Firewall-Architekturen für Multimedia-Applikationen

vom Fachbereich 20
der Technischen Universität Darmstadt
genehmigte Dissertation
zur Erlangung des Grades eines
Doktor-Ingenieurs (Dr.-Ing.)

von

Dipl.-Ing. Utz Roedig
geboren am 6.7.1972 in Offenbach

Darmstadt 2002
Hochschulkennziffer D17

Vorsitzender:	Prof. Alejandro Buchmann, Ph.D.
Referent:	Prof. Dr.-Ing. Ralf Steinmetz
Korreferent:	Prof. Dr. techn. Joachim Swoboda

Tag der Einreichung:	15. Oktober 2002
Tag der Disputation:	29. November 2002

Eidesstattliche Erklärung

Ich versichere hiermit an Eides statt, dass ich die vorliegende Arbeit allein und nur unter Verwendung der angegebenen Literatur verfasst habe.

Darmstadt, den 14. Oktober 2002

_____

Dipl.-Ing. Utz Roedig

Kurzfassung

In der vorliegenden Arbeit “Firewall-Architekturen für Multimedia-Applikationen” werden Lösungen entwickelt und diskutiert, die es ermöglichen, Multimedia-Applikationen in Netzwerken zu verwenden, in denen Firewalls eingesetzt werden. Die dargestellten Lösungen decken die Optimierung bestehender Firewall-Architekturen sowie die Entwicklung neuer Mechanismen für die Realisierung von Firewall-Architekturen ab.

In einer immer vernetzteren Umgebung nimmt die Bedeutung von Sicherheitsaspekten stetig zu und eine Zugangskontrolle an Netzgrenzen wird als wesentlich betrachtet. Zu diesem Zweck werden Firewalls verwendet, die diese Zugangskontrolle an der Grenze zwischen offenen und privaten Netzen bereitstellen. Als integraler Bestandteil der Netzwerkinfrastruktur werden Firewalls stark durch die Entwicklung und den Einsatz neuer Kommunikationsformen und Applikationen beeinflusst. Zur Zeit kann beobachtet werden, dass Multimedia-Applikationen, die sich in vielerlei Hinsicht von “traditionellen” Applikationen unterscheiden, zunehmend Verwendung finden. Existierende Firewalls sind nicht in der Lage, diese neuen Applikationstypen in effizienter und sicherer Weise zu unterstützen.

In der vorliegenden Arbeit werden die bestehenden Problembereiche identifiziert und klassifiziert. Ausgehend von dieser Klassifizierung wird gefolgert, dass die Veränderung und Erweiterung bestehender Firewall-Architekturen eine geeignete Maßnahme zur Lösung dieser Probleme darstellt. Es wird gezeigt, dass eine geeignete Architektur dem in der Arbeit vorgestellten Designprinzip der Trennung von Signalisierungs- und Medienströmen folgen muss.

Es wird ein Architekturmodell vorgestellt, das Firewall-Architekturen hinsichtlich der für die Verarbeitung von Multimedia-Applikation relevanten Architekturmerkmale ordnet. Die dadurch mögliche Betrachtung verschiedener Architektur-Klassen zeigt, dass die Klasse der verteilten Firewall-Architekturen für einen Einsatz im Multimedia-Umfeld geeignet ist. Das in dieser Arbeit vorgestellte Modell ermöglicht die Identifikation der fehlenden bzw. zu optimierenden Elemente, damit eine verteilte Firewall-Architektur effizient realisiert werden kann.

Ein wesentliches, zur Umsetzung einer verteilten Firewall notwendiges Element ist die Kommunikation zwischen den einzelnen Firewall-Komponenten. Anstatt für diese Kommunikation ein neues Protokoll zu entwerfen - wie zur Zeit in verschiedenen Standardisierungsgremien vorgeschlagen wird - wird in der Arbeit gezeigt, dass das bestehende und erprobte Resource Reservation Protocol (RSVP) dafür verwendet werden kann. Anhand einer Implementierung wird gezeigt, dass das RSVP auch in der Praxis für diese Aufgabe eingesetzt werden kann.

Ein weiteres notwendiges Element zur Umsetzung einer verteilten Firewall stellt die Signalisierungsverarbeitung dar. Bisher verwendete Methoden, die Signalisierungsverarbeitung der

Firewall in ein Kommunikationsszenario einzubinden, können nicht auf Multimedia-Szenarien übertragen werden. In der Arbeit wird gezeigt, dass die für die Integration der Signalisierungsverarbeitung notwendigen Mechanismen von bestehenden Integrationsmechanismen für Infrastrukturkomponenten abgeleitet werden müssen. Durch eine Implementierung wird gezeigt, dass diese Integrationsmechanismen auch in der Praxis anwendbar sind.

Besonderes Augenmerk wird in der Arbeit auf die Leistungsfähigkeit von Firewall-Architekturen gerichtet. Es wird dargestellt, welche Faktoren wesentlich die mögliche Leistungsgrenze einer Multimedia-Firewall bestimmen und wie diese zu berücksichtigen sind, um eine Multimedia-Firewall für bestimmte Leistungsanforderungen richtig zu dimensionieren. Anhand von Messungen verschiedener Firewall-Architekturen wird abschliessend gezeigt, dass die innerhalb der Arbeit vorgestellten Architekturvorschläge auch hinsichtlich ihrer Leistungsfähigkeit für die Unterstützung von Multimedia-Applikationen geeignet sind. Verteilte Firewall-Architekturen, die das aufgestellte Designprinzip der Trennung von Signalisierungs- und Medienströmen berücksichtigen, müssen verwendet werden, um die Leistungsgrenze einer Firewall zu optimieren.

Im Rahmen dieser Arbeit wurden zwei Werkzeuge entwickelt, die zur Überprüfung der getroffenen Aussagen verwendet wurden, aber auch über die Arbeit hinausgehend verwendet werden können. Das Werkzeug *KOMtraffgen* stellt ein Messwerkzeug zur Bestimmung von Leistungskenngrößen dar. Es kann verwendet werden, um Leistungskenngrößen von Komponenten im Kommunikationspfad einer Multimedia-Applikation zu bestimmen. Das Werkzeug *KOMproxyd* kann verwendet werden, um Firewall-Architekturen für Multimedia-Applikationen umzusetzen. Dieses Werkzeug wird beispielsweise durch das Deutsche Forschungsnetz (DFN) innerhalb des DFN-Videokonferenzdienstes zur Zeit in der Praxis verwendet.

Abstract

In this thesis on *Firewall Architectures for Multimedia Applications* solutions are developed and discussed that enable the usage of multimedia applications in network environments where firewalls are employed. The provided solutions cover optimizations of existing firewall architectures as well as the development of new mechanisms to implement firewall architectures.

Within a global networked environment, security aspects become more and more important and access control at network borders is considered to be essential. For this purpose firewalls which provide access control and auditing at the border between open and private networks or administrative domains are used. As integral part of the network infrastructure they are strongly affected by the development and deployment of new communication paradigms and applications. Currently we experience a very fast rise in the use of multimedia applications which differ in many aspects from “traditional” applications. Existing firewalls are not able to support this new types of applications in an efficient and secure manner.

This thesis identifies and classifies the existing problem areas. It can be deduced from this classification that a modification and extension of existing firewall architectures are suitable methods to solve these problems. In the thesis it is shown that an appropriate firewall architecture has to apply the design pattern “Separation of Signalling and Media Flows”.

A new architectural model is introduced which can be used to structure firewall architectures regarding the criterias necessary to support multimedia applications. Thus, it is possible to investigate different categories of architectures and it is shown that the category of distributed firewalls fits best to support multimedia applications. This model also allows to identify which elements are missing or have to be optimized to build distributed firewalls.

An important element of a distributed firewall is the communication between the different firewall components. Instead of developing a new protocol - as currently proposed in the standardization bodies - it is shown that the existing and approved Resource Reservation Protocol (RSVP) can be used for this purpose. It is shown by an implementation that RSVP can be used in practice.

Another important element used in firewall architectures is the signalling element. State of the art methods used for integration of the signalling element within a scenario cannot be used in multimedia scenarios. It is shown in the thesis that the necessary integration mechanisms have to be deduced from integration mechanisms used for multimedia infrastructure components. On the basis of an implementation it is shown that this approach is also feasible in practice.

Within the thesis the performance of firewall architectures is investigated. The parameters which limit the performance of a multimedia firewall are identified. It is shown how these parameters have to be taken into account to optimize a firewall for specific performance requirements. Mea-

surements are performed to show that the proposed changes in firewall architectures are optimal regarding the performance. Distributed firewalls that use the design pattern “Separation of Signalling and Media Flows” have to be used to optimize the performance of a multimedia firewall.

Within the thesis several tools had been developed to show the feasibility of the given statements which can also be used for other purposes not regarded in this thesis. The tool *KOMtraffgen* can be used for performance measurements as well as to determine performance values of components used in the communication path of multimedia applications. The tool *KOMproxyd* can be used to build firewall architectures for multimedia applications. It is currently used within the video conference service of the Deutsches Forschungsnetz (DFN).

Inhaltsverzeichnis

Abbildungsverzeichnis

Kapitel 1: Einleitung

Verteilte multimediale Dienste, die das Internet als Transportmedium verwenden, werden heute in zunehmendem Maß genutzt. Als bekannte Beispiele für solche multimedialen Dienste können Videokonferenz-, E-Learning-, Internet-Telefonie (VoIP) oder Video on Demand (VoD) Dienste genannt werden. Je nach Standpunkt sind die unterschiedlichsten Erwartungen an die Verwendung dieser Dienste geknüpft. Manchmal wird ein solcher Dienst als kostengünstiger Ersatz für eine bestehende Lösung gesehen (z.B. Ersatz der klassischen Telefonie durch Internet-Telefonie), oder aber man verspricht sich durch den Einsatz eines Dienstes einen gewissen Mehrwert gegenüber anderen bestehenden Angeboten (z.B. Video on Demand gegenüber dem klassischen Fernsehen).

Soll ein Dienst kommerziell, also über ein reines Versuchsstadium hinaus, verwendet werden, so ist die Sicherheit des Dienstes meist ein wesentliches Kriterium für seinen Erfolg. Ein Anwender wird einen Dienst erst dann nutzen und auch dafür bezahlen, wenn neben dem Nutzen und der Qualität dieser auch seinen Schutzbedürfnissen in geeignetem Umfang nachkommt. Ebenso wird ein Betreiber einen Dienst nur dann anbieten, wenn einerseits die Sicherheit das Erreichen seiner Geschäftsziele ermöglicht und andererseits die Sicherheit anderer Dienste nicht beeinträchtigt wird.

Es hat sich gezeigt, dass der sichere Betrieb von multimedialen Diensten zur Zeit noch nicht in ausreichendem Maß möglich ist. Dies liegt zum einen an der mangelhaften Umsetzung vorhandener Sicherheitsmechanismen, zum anderen daran, dass passende Methoden und Lösungen noch nicht existieren. Diese Methoden und Lösungen zu entwickeln ist Aufgabe der Wissenschaft und Bestandteil dieser Arbeit.

1.1 Motivation

Um den verschiedenartigsten Schutzbedürfnissen nachkommen zu können, werden unterschiedliche Maßnahmen ergriffen. Eine Firewall ist eine bekannte und häufig eingesetzte Maßnahme zur Abwehr bestimmter Bedrohungen. Sie ermöglicht es, an einem Übergangspunkt zwischen zwei, meist unter getrennter administrativer Verwaltung stehender Netzwerke, Sicherheitsüberprüfungen von über die Netzgrenze fließenden Daten vorzunehmen. Werden multimediale Dienste dort verwendet, wo Firewalls als Schutzmechanismus verwendet werden, treten Probleme auf, die durch das folgende Beispiel verdeutlicht werden: Viele Radiostationen bieten alternativ zu ihrem über die konventionellen Kanäle ausgestrahlten Radioprogramm mittlerweile auch ein so genanntes Web-Radio an. Dieser Dienst ermöglicht es dem Hörer ein Radioprogramm live über das Internet zu empfangen. Der Hörer benötigt dazu eine passende Applikation, die die Audiodaten von

einem Server der Radiostation bezieht. Ein Hörer in den USA ist so zum Beispiel in der Lage, über das Internet das Programm eines deutschen Radiosenders zu empfangen. In der Praxis tritt aber nun häufig das Problem auf, dass der Hörer sich in einem durch eine Firewall geschützen Netzwerkbereich befindet. Die Firewall ist nicht fähig, die der Übertragung des Programms zugrunde liegende Kommunikation zu unterstützen, was die Nutzung des Dienstes verhindert. Um dieses Problem zu lösen, werden folgende Ansätze verwendet: Zum einen kann die Firewall durch Einschränkung ihrer Schutzfunktion so betrieben werden, dass die Radioübertragung möglich wird. Zum anderen kann die zur Übertragung genutzte Kommunikationsform geändert werden. Diese kann dann durch die Firewall unterstützt werden, eignet sich aber eigentlich nicht für eine Audioübertragung. Beide Lösungen sind nicht wünschenswert, da die eine Lösung die Sicherheit beeinträchtigt, die andere die Qualität des Dienstes.

Die beschriebene Beobachtung kann ebenfalls bei der Verwendung anderer Multimedia-Applikationen (bzw. multimedialen Diensten) gemacht werden. Die auftretenden Probleme sind allgemeinerer Natur und nicht auf wenige Spezialfälle beschränkt. Damit multimediale Dienste erfolgreich eingesetzt werden können, muss der beschriebene Problemkomplex des Zusammenwirkens von Multimedia-Applikationen und Firewalls gelöst werden. Es müssen passende Methoden entwickelt werden, die einen gemeinsamen, uneingeschränkten und sicheren Betrieb von Multimedia-Applikationen zusammen mit Firewalls ermöglichen.

1.2 Zielsetzung

Ziel dieser Arbeit ist es, Methoden zu entwickeln, die ein problemloses Zusammenspiel von Multimedia-Applikationen mit dem Schutzmechanismus Firewall ermöglichen.

Um dieses Ziel erreichen zu können, ist es als Erstes notwendig, die Ursachen der bestehenden Probleme zu ergründen. Das erste Teilziel dieser Arbeit besteht deshalb darin, die Probleme zu analysieren und hinsichtlich ihrer Ursachen zu strukturieren. Davon ausgehend können dann Lösungsmechanismen für diese Probleme gefunden werden.

Die Aufgaben, die eine Firewall zu erfüllen hat, werden in dieser Arbeit als fest vorgegebene Rahmenbedingung betrachtet. Es ist dabei allerdings offen, welche Firewall-Architektur gewählt wird. Das zweite Teilziel dieser Arbeit ist daher Mechanismen für die Realisierung von Firewall-Architekturen zu finden, die den uneingeschränkten Betrieb einer Firewall in einem Multimedia-Umfeld ermöglichen. Im Vordergrund steht dabei die Identifikation nötiger Architekturelemente. Es ist zu prüfen, ob Mechanismen für die Umsetzung dieser Elemente bereits existieren, bestehende Mechanismen angepasst oder optimiert verwendet werden können oder neue Mechanismen benötigt werden. Die so geschaffenen bzw. verbesserten Mechanismen müssen abschließend hinsichtlich ihrer Eignung bewertet werden. Dies schließt insbesondere die im Multimedia-Umfeld wichtige Bewertung der Leistungsfähigkeit ein.

1.3 Struktur der Arbeit

In Kapitel 2 dieser Arbeit werden die für das Verständnis der Arbeit notwendigen Definitionen gegeben. Es werden die notwendigen Teilbereiche der Themengebiete “Sicherheit”, “Firewalls” und “Multimedia-Applikationen” identifiziert und erläutert.

Auf Probleme, die sich bei der Integration von Multimedia-Applikationen in durch Firewalls geschützten Netzwerken ergeben, wird in Kapitel 3 ausführlich eingegangen. Es werden die Ursachen dieser Probleme beschrieben und eine Klassifikation derselben wird vorgenommen. Anhand dieser Klassifikation werden Lösungsansätze erarbeitet, von denen wiederum die Anforderungen abgeleitet werden können, die geeignete Firewall-Architekturen erfüllen müssen.

Kapitel 4 stellt ein Architekturmodell vor, das Firewall-Architekturen hinsichtlich der für die Verarbeitung von Multimedia-Applikationen relevanten Architekturmerkmale ordnet. Das Modell identifiziert Gemeinsamkeiten, bzw. die Verwendung identischer Elemente, innerhalb verschiedener Architekturklassen. So werden die für die Umsetzung von Multimedia-Applikationen vorhandenen sinnvollen, bzw. noch fehlenden Architekturelemente ermittelt. Dieses Kapitel setzt sich auch mit anderen relevanten Arbeiten auf dem Gebiet der Multimedia-Firewalls auseinander.

Für die Umsetzung einer verteilten Firewall-Architektur, die sich für ein Multimedia-Umfeld besonders eignet, sind passende Kommunikationsmechanismen notwendig. Für die Kommunikation zwischen den Firewall-Komponenten wird ein sogenanntes Firewall Control Protocol (FCP) verwendet. Kapitel 5 zeigt, wie dieses Architekturelement effizient durch die Verwendung bestehender Elemente des Resource Reservation Protocols (RSVP) umgesetzt werden kann.

In Kapitel 6 wird dargestellt, wie das Architekturelement “Signalisierungsverarbeitung” für eine Multimedia-Firewall umgesetzt werden muss. Außerdem wird festgestellt, dass bisher verwendete Konzepte zur Umsetzung der Signalisierungsverarbeitung im Multimedia-Umfeld ungeeignet sind.

Kapitel 7 untersucht die Leistungsfähigkeit von Firewall-Architekturen. Um den strikten Dienstgüteanforderungen einer Multimedia-Applikation gerecht zu werden, muss eine Firewall bestimmte Leistungsanforderungen erfüllen. In diesem Kapitel wird außerdem gezeigt, dass die in den vorhergehenden Kapiteln erarbeiteten Vorschläge auch unter dem Gesichtspunkt der Leistungsfähigkeit optimierte Lösungen darstellen.

Kapitel 8 fasst die in dieser Arbeit gewonnenen Ergebnisse zusammen und gibt einen Ausblick auf das in der Arbeit behandelte Themengebiet “Firewall-Architekturen für Multimedia-Applikationen”.

Kapitel 2: Grundlagen

In diesem Kapitel werden die Grundlagen, die zum Verständnis der Arbeit notwendig sind, erläutert. Es werden die notwendigen Teilbereiche der Themengebiete “Sicherheit”, “Firewalls” und “Multimedia-Applikationen” identifiziert und erklärt. Für die wesentlichen Begriffe und Konzepte werden jeweils Definitionen gegeben, insbesondere, wenn aus der Literatur keine oder nicht eindeutige Definitionen bekannt sind. Eine umfangreiche Darstellung der einzelnen Themengebiete kann und soll in dieser Arbeit nicht erfolgen, weshalb an den entsprechenden Stellen auf weiterführende Literatur verwiesen wird.

2.1 IT-Sicherheit

Das Themengebiet “IT-Sicherheit” beinhaltet neben den technischen Aspekten weitere wesentliche, aber nicht technische Aspekte. Dazu gehören beispielsweise juristische [1], psychologische [2] oder wirtschaftliche [3] Fragestellungen. Ein Großteil der technischen Aspekte kann mit Mitteln der Informationstechnik bearbeitet werden. Auf diesen Aspekten der IT-Sicherheit liegt der Fokus dieser Arbeit. Entsprechend dieser Abgrenzung wird das Themengebiet “IT-Sicherheit” im Folgenden verstanden.

Das resultierende Themengebiet deckt noch ein sehr großes Feld ab, wodurch eine weiterführende Strukturierung notwendig wird. Bestehende Literatur verwendet hierfür verschiedene Möglichkeiten: Aufteilung entsprechend der Kommunikationsschichten [4], Aufteilung nach Angriffsmethoden [5], Aufteilung nach Maßnahmen [6], usw.. Da keine allgemeingültige Strukturierung des Gebietes “IT-Sicherheit” gegeben ist, können für einen zu betrachtenden Aspekt mehrere passende Strukturierungen gewählt werden. In den einzelnen gewählten Strukturierungen können dann die wesentlichen Teilbereiche für den zu betrachtenden Aspekt identifiziert werden. Für den Aspekt “Firewall-Architekturen für Multimedia-Applikationen” bieten sich folgende drei Strukturierungen an:

- **Aufteilung entsprechend der Kommunikationsschichten:** Eine Firewall wird als Baustein innerhalb einer Netzinfrastruktur verwendet. Wird eine Aufteilung entsprechend der Kommunikationsschichten (z.B. Applikation, Netzwerk) gewählt, so liegt der Fokus dieser Arbeit innerhalb des Themenbereichs “Netzwerksicherheit”. Der Aspekt “Netzwerksicherheit” wird in [4] und [7] allgemein beschrieben.
- **Aufteilung nach Maßnahmen:** Nimmt man eine Unterteilung der IT-Sicherheit nach möglichen Schutzmaßnahmen (z.B. Virtuelle Private Netze (VPNs), Firewalls, verschlüsselte File-Systeme) vor, so liegt der Fokus dieser Arbeit auf dem unter dem

Begriff “Firewalls” zusammengefassten Themenbereich. Dieser Aspekt der IT-Sicherheit wird beispielsweise in [8], [6], [9] und [10] allgemein beschrieben.

- **Aufteilung nach Applikationstypen:** Werden als Unterteilung die verschiedenen zu schützenden Applikationstypen herangezogen, so liegt der Fokus dieser Arbeit auf dem Gebiet der “Sicherheit von verteilten Multimedia-Anwendungen”. Dieser Aspekt wird unter Anderem in [11] behandelt.

Für die im Folgenden behandelte Thematik “Firewall-Architekturen für Multimedia-Applikationen” stellt die Schnittmenge der oben hervorgehobenen Themengebiete den in der Arbeit berücksichtigten Teil der IT-Sicherheit dar. Dieser Teil der IT-Sicherheit kann wie folgt graphisch dargestellt werden:

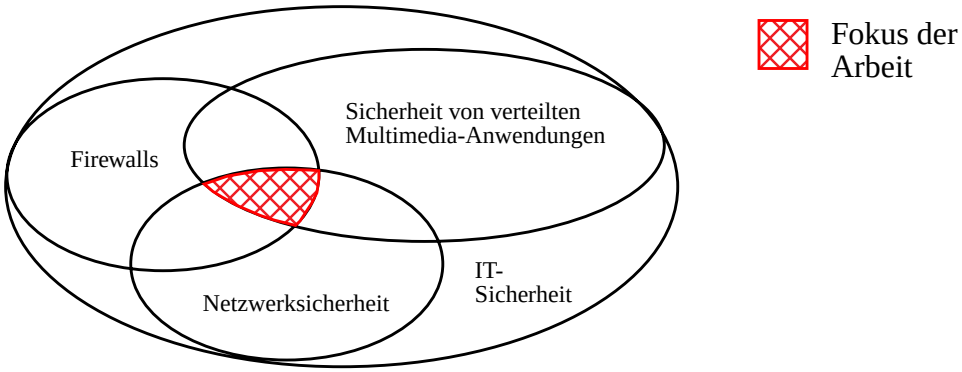


Abbildung 1: Fokus der Arbeit

2.1.1 Wechselwirkung zwischen Systemanforderungen

An ein IT-System werden die unterschiedlichsten Anforderungen gestellt, die seine Nützlichkeit sicherstellen oder erhöhen sollen. An erster Stelle stehen dabei zunächst Anforderungen hinsichtlich der Funktionalität. Die Tatsache, dass ein IT-System ebenfalls Anforderungen hinsichtlich der Sicherheit gerecht werden muss, wird allerdings aus verschiedenen Gründen oft nicht genügend berücksichtigt. Ein wesentlicher Grund dafür ist, dass Sicherheit eine nicht-funktionale Eigenschaft ist. Ein Benutzer, dessen System korrekt funktioniert, kann eigentlich nicht sagen, ob es auch sicher ist[1]. Dennoch ist für den letztendlichen Erfolg eines IT-Systems auch dessen Sicherheit von Bedeutung, da diese Einfluss auf die Erfüllung der primären funktionalen Anforderungen hat. Zusätzlich hat der Sicherheitsaspekt erheblichen Einfluss auf nicht-technische Anforderungen wie z.B. die Wirtschaftlichkeit oder die Gesetzeskonformität der IT-Systeme.

Insbesondere die zweitrangige Betrachtung der Sicherheit führt dazu, dass IT-Systeme erst nachträglich in ein Sicherheitskonzept integriert werden, bzw. ein Sicherheitskonzept für sie erstellt wird. Dies führt in der Regel dazu, dass Sicherheitsanforderungen nicht mehr in dem Maße umgesetzt werden können, wie dies bei einer Beachtung dieser Anforderungen schon bei

[1] Das Testen eines Systems kann immer nur das Vorhandensein von Fehlern zeigen, aber niemals die Fehlerfreiheit. Diese Aussage lässt sich auf das Gebiet der IT-Sicherheit übertragen, wenn man ein fehlerhaftes System als unsicheres System versteht.

der Konzeption des IT-Systems möglich gewesen wäre. Diese Problematik kann durch verschiedene Beispiele belegt werden. Das für die IP-Telefonie verwendete H.323-Protokoll [12], und damit auch die darauf aufbauenden IP-Telefoniesysteme, wurde zunächst hinsichtlich der Anforderung Funktionalität entwickelt. Sicherheitsanforderungen spielten bei der Entwicklung der ersten Version des H.323-Protokolls so gut wie keine Rolle. Dadurch ergeben sich beispielsweise folgende Probleme:

- Die nachträgliche Integration von Sicherheitsanforderungen innerhalb der H.323-IT-Systeme ist nur unter Berücksichtigung der schon zuvor festgelegten Gegebenheiten möglich. Es muss eine Authentifizierung von Gesprächsteilnehmern in die gegebene H.323-Signalisierung integriert werden, obwohl diese für diesen Zweck nicht optimal ausgelegt ist. Dass dies ein Problem darstellt, zeigt sich an den zahlreichen Arbeiten auf diesem Gebiet [13], [14], [15], [16], [17].
- Die nachträgliche Anpassung heute verwendeter H.323-IT-Systeme an gegebene Sicherheitsanforderungen ist nur möglich, wenn die gegebenen Sicherheitsanforderungen selbst verändert werden. Die durch eine Firewall umgesetzten Sicherheitsanforderungen müssen eingeschränkt werden, damit die notwendigen H.323-IT-Systemanforderungen erfüllt werden können.

Es zeigt sich, dass Wechselwirkungen zwischen Sicherheitsanforderungen und anderen Systemanforderungen bestehen. Daraus folgt, dass eine optimale (eine ausreichende Umsetzung aller Anforderungen) Gesamtlösung in der Regel nur dann erreicht werden kann, wenn alle Parameter (Sicherheitsmechanismen sowie andere Systemparameter) gemeinsam angepasst werden.

Dies bedeutet zum einen für die vorliegende Arbeit, dass sowohl Sicherheitsanforderungen, die in Firewalls umgesetzt werden, an die Systemanforderungen von Multimedia-Systemen angepasst werden müssen. Zum anderen aber auch, dass Multimedia-Systeme ebenfalls angepasst werden müssen, um keine funktionalen Einschränkungen durch die in einer Firewall umgesetzten Sicherheitsanforderungen zu erfahren. Insbesondere der zweite Aspekt kann in der Praxis nur schwer umgesetzt werden, da eine Neudefinition bestehender Systemarchitekturen meist nicht möglich ist.

2.1.2 Definitionen

Im Folgenden werden die aus dem Bereich der IT-Sicherheit verwendeten Begriffe, die innerhalb dieser Arbeit benötigt werden, definiert. Eine Beschreibung dieser Begriffe ist notwendig, da in der Literatur uneinheitliche Definitionen existieren.

Der Begriff “Sicherheit” wird in den unterschiedlichsten Bereichen verwendet. Eine allgemeine Definition dieses Begriffes ist in [18] folgendermaßen gegeben:

> ***Sicherheit**, Zustand des Unbedrohtseins, der sich objektiv im Vorhandensein von Schutz[einrichtungen] bzw. im Fehlen von Gefahr[enquellen] darstellt und subjektiv als Gewissheit von Individuen oder sozialen Gebilden über die Zuverlässigkeit von Sicherheits- und Schutzeinrichtungen empfunden wird.*

Durch das am Beginn dieses Kapitels beschriebene Verständnis des Themenfeldes IT-Sicherheit ist deutlich, dass eine solche Definition hier zu allgemein ist. Deshalb wird in dieser Arbeit die folgende Definition des Begriffs “Sicherheit” angelehnt an [19] verwendet:

> ***Sicherheit*** *von IT-Systemen ist der Schutz von Verfügbarkeit, Vertraulichkeit und Integrität.*
>
> ***Verfügbarkeit*** *bedeutet, dass ein Informationssystem im vorgesehenen zeitlichen Rahmen erreichbar und nutzbar ist.*
>
> ***Vertraulichkeit*** *bedeutet die Freigabe von Daten und Informationen nur an autorisierte Personen, Gruppen und Prozesse zu einer vorgegebenen Zeit und in einer vorherbestimmten Art und Weise.*
>
> ***Integrität*** *bedeutet, dass Daten und Informationen vollständig und genau sind und dass die Vollständigkeit und Genauigkeit über die Zeit erhalten bleiben.*

Die Sicherheit eines IT-Systems wiederum ist verschiedenen Bedrohungen ausgesetzt. In Anlehnung an [20] kann der Begriff “Bedrohung” folgendermaßen beschrieben werden:

> *Eine* ***Bedrohung*** *tritt dann auf, wenn es entsprechende Umstände, Möglichkeiten, Aktionen oder Ereignisse gibt, die die Sicherheit gefährden.*

Eine mögliche Bedrohung stellen Angriffe [20] auf IT-Systeme dar.

> *Durch einen* ***Angriff*** *versucht ein Angreifer, die Sicherheit eines IT-Systems zu unterwandern und zu seinem Vorteil auszunutzen.*

Eine spezielle Form des Angriffs ist ein Denial of Service (DoS) Angriff. Ziel eines DoS-Angriffs ist es, ein IT-System unbenutzbar zu machen. Ein solcher Angriff richtet sich gegen die Verfügbarkeit eines IT-Systems.

Ein Angriff kann nur dann erfolgreich ausgeführt werden, wenn das angegriffene System eine Verletzbarkeit aufweist.

> *Ein System weist eine* ***Verletzbarkeit*** *auf, wenn es nur in einem unzureichenden Maß Schutz gegen Missbrauch gewährleistet.*

Wird diese Verletzbarkeit durch einen Angreifer ausgenutzt, so ist die Sicherheit des betroffenen Systems gefährdet. Es gibt eine Reihe alternativer Strategien, um einen Schutz vor Angriffen zu erreichen. Diese Strategien können verwendet werden, um vorhandene Bedrohungen zu neutralisieren.

> *Als* ***Maßnahmen*** *werden die verschiedenen technischen Umsetzungen von Strategien zum Schutz vor Angriffen bezeichnet. Es ergänzen sich dabei proaktive und reaktive Maßnahmen.*
>
> ***Proaktive Maßnahmen*** *richten sich gegen Bedrohungen, die vorhergesehen wurden und gegen die bereits im Vorfeld Vorkehrungen getroffen werden können.*
>
> ***Reaktive Maßnahmen*** *zielen darauf ab, einen Angriff als solchen zu erkennen und Gegenmaßnahmen einzuleiten. Nur wenn ein Angriff erkannt worden ist, kann korrigierend eingegriffen werden, d.h. es werden entsprechende Schritte als Reaktion auf den Angriff eingeleitet.*

Primäres Ziel aller Maßnahmen ist es, die Verletzbarkeiten oder aber die Auswirkungen von Verletzbarkeiten zu eliminieren.

2.2 Firewalls

Eine Firewall ist eine von vielen möglichen Maßnahmen zum Schutz vor speziellen Bedrohungen. In erster Linie handelt es sich bei einer Firewall um eine proaktive Maßnahme zum Schutz vor Bedrohungen. Eine Firewall ermöglicht es, an einem Übergangspunkt zwischen zwei, meist unter getrennter administrativer Verwaltung stehender Netzwerke, Überprüfungen und/oder Modifikation der über die Netzgrenze fließenden Daten anhand der gegebenen Sicherheitsanforderungen vorzunehmen. Dementsprechend kann eine Firewall nur als Maßnahme gegen Bedrohungen eingesetzt werden, die an dem entsprechenden Netzübergang, an dem die Firewall eingesetzt wird, neutralisiert werden können. Eine Firewall kann also nicht die alleinige Maßnahme zur Beseitigung aller Bedrohungen darstellen und muss durch andere Maßnahmen ergänzt werden. Grundlegende Beschreibungen zur Funktionsweise einer Firewall finden sich beispielsweise in [9] und [10].

2.2.1 Definitionen

Im Folgenden werden zwei sich ergänzende Definitionen einer Firewall gegeben. Die erste Definition beschreibt die durch eine Firewall zu übernehmenden Aufgaben, die zweite Definition beschreibt die Funktionsweise einer Firewall.

Definition der Aufgaben. Eine Firewall implementiert bestimmte Funktionen, um bestimmte Bedrohungen zu neutralisieren. In der Literatur herrscht dabei Uneinigkeit, welche Aufgaben bzw. Funktionen genau von einer Firewall übernommen werden bzw. dieser zugerechnet werden müssen. Im Wesentlichen können alle Maßnahmen, die der Vermeidung von Bedrohungen an einer Netzgrenze dienen, einer Firewall zugesprochen werden. In der vorliegenden Arbeit werden die im Folgenden beschriebenen Aufgaben von einer Firewall übernommen, wodurch sich eine Definition des Begriffs *Firewall* ergibt:

> *Eine **Firewall** übernimmt folgende Aufgaben: Identitätsbezogene Zugriffskontrolle, Filterung und Modifikation von Daten, Verbergen von Strukturen sowie Audit sicherheitsrelevanter Ereignisse.*
>
> ***Zugriffskontrolle:** Die Firewall muss in der Lage sein, basierend auf den Identitäten von Sender und Empfänger der über die Firewall laufenden Daten, eine Zugriffskontrolle durchzuführen. Die Identitäten können durch die Analyse der Daten, oder durch andere Verfahren ermittelt werden. Beispielsweise kann als Identität eine Benutzerkennung, ein bestimmter Prozess auf einem Rechner oder ein Rechner verwendet werden.*
>
> ***Filtern und Modifizieren:** Basierend auf den durch die Analyse der über die Firewall laufenden Daten gewonnenen Informationen muss entschieden werden, ob durch die übermittelten Daten eine Bedrohung entstehen kann oder nicht. Aufgrund dieser Entscheidung werden die Daten dann weitergeleitet oder verworfen. Zusätzlich kann festgelegt werden, ob*

Daten vor einer Weiterleitung modifiziert werden müssen, um erkannte Bedrohungen zu neutralisieren.

Verbergen: *Eine Firewall muss in der Lage sein, die auf der einen Seite der Netzgrenze liegenden Strukturen vor der anderen Seite zu verbergen. Dies entzieht einem Angreifer die Möglichkeit, sich Informationen über mögliche Ziele zu beschaffen.*

Audit: *Angriffe auf IT-Systeme bleiben unentdeckt, wenn es keine Möglichkeit gibt Unregelmässigkeiten im Betrieb festzustellen. Eine Firewall muss deshalb in der Lage sein, für die oben beschriebenen Funktionen ein Audit durchzuführen.*

Die in der obenstehenden Definition festgelegten Kernfunktionen einer Firewall stellen ebenfalls die wesentlichen Bereiche dar, in denen es zu Problemen bei einem Datenaustausch von Multimedia-Applikationen über eine Firewall kommt (siehe Kapitel 3). Aus diesem Grund eignet sich die gegebene Definition für die vorliegende Arbeit.

Definition der Funktionsweise. Die Funktionsweise einer Firewall kann durch das in Abbildung 2 dargestellte Modell in Anlehnung an [21] beschrieben werden. Abbildung 2 zeigt die einzelnen Elemente - Analysator, Entscheider, Umsetzer, Regelspeicher und Audit-Element - einer Firewall, sowie deren Kommunikationsbeziehungen.

Ein Sender übermittelt ein Protokollelement X_i aus der Menge X der möglichen Protokollelemente an den Empfänger. Wird das Protokollelement durch den Empfänger verarbeitet, führt dieser eine Aktion A_i aus der Menge A der möglichen Aktionen aus. Wird eine Firewall in den Kommunikationsweg zwischen Sender und Empfänger eingebracht, so wird das Protokollelement vor dem Weiterleiten an den Empfänger zunächst durch die Firewall verarbeitet. X_i wird analysiert und der Entscheider ermittelt dann anhand der vorliegenden Regeln sowie dem Analyseergebnis, welche Aktionen vorzunehmen sind. Die entsprechenden auszuführenden Aktionen ru_i und ra_i aus der Menge der möglichen Aktionen *ru* und *ra* werden festgelegt. Die auszuführenden Aktionen werden an das jeweils zuständige Element übermittelt. Der Umsetzer ist nun in der Lage, X_i weiterzuleiten, zu verwerfen oder zu modifizieren. Dadurch wird erreicht, dass bestimmte nicht gewünschte - festgelegt durch die Regeln - Aktionen $A^* \subset A$ vom Empfänger nicht ausgeführt werden.

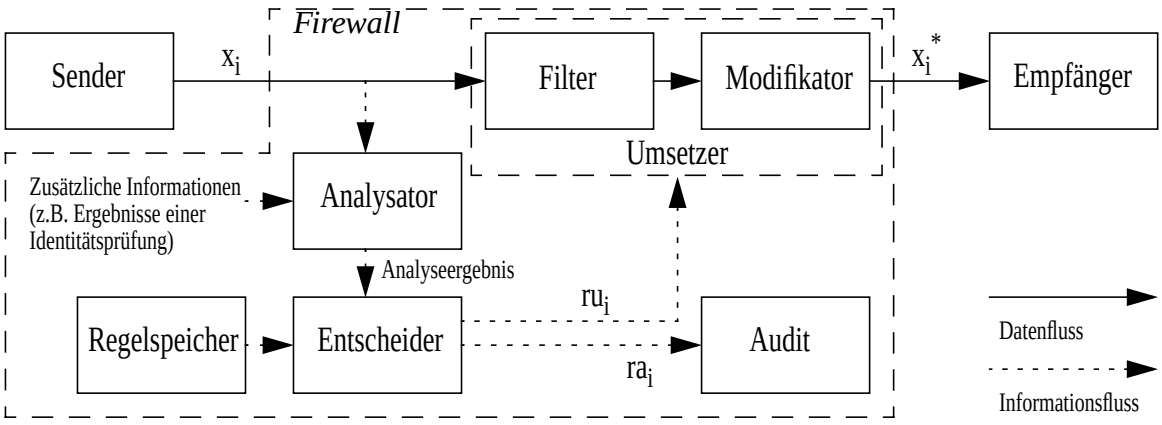


Abbildung 2: Funktionsmodell einer Firewall

Die durch die einzelnen Elemente einer Firewall erbrachten Funktionen sind folgendermaßen festgelegt:

> ***Analysator:*** *Die Firewall muss in der Lage sein, die über sie fließenden Daten zu analysieren. Die durch die Analyse gewonnenen Informationen werden zur Realisierung der Aufgaben einer Firewall benötigt. Die Analyse der Daten kann auf verschiedene Arten erfolgen, die sich hinsichtlich ihrer Parameter wie z.B. technische Umsetzung, Güte oder Granularität unterscheiden. Beispielsweise ist eine Analyse der Semantik der Header einzelner Protokolle, die Analyse der durchlaufenden Zustände einer Protokollzustandsmaschine, die zwischen zwei Kommunikationspartnern abgewickelt wird, oder die Analyse der Zusammenhänge verschiedener parallel aktiver Kommunikationsverbindungen denkbar.*
>
> ***Entscheider:*** *Das Resultat der Anlalyse wird verwendet, um anhand der Regeln festzustellen, welche Aktionen auszuführen sind. Die auszuführenden Aktionen werden an die dafür zuständigen Elemente übermittelt.*
>
> ***Regelspeicher:*** *Innerhalb des Regelspeichers sind die einzelnen Regeln der Firewall abgelegt. Die* ***Regeln*** *beschreiben, wie das jeweilige Analyseergebnis durch den Entscheider bewertet wird. Eine Regel besteht aus zwei Teilen. Der erste Teil beschreibt, welche Bedingung erfüllt sein muss, damit die im zweiten Teil der Regel festgelegte Aktion ausgeführt wird.*
>
> ***Audit:*** *Entsprechend der durch den Entscheider festgelegten Aktion ra_i wird ein Audit durchgeführt.*
>
> ***Umsetzer:*** *Entsprechend der durch den Entscheider festgelegten Aktion ru_i wird das Protokollelement X_i durch den Umsetzer behandelt. Der* ***Filter*** *kann X_i verwerfen oder weiterleiten. Der nachfolgend geschaltete* ***Modifikator*** *kann X_i modifizieren. Eine Modifikation ist beispielsweise nötig, um Bedrohungen zu neutralisieren oder ein Verbergen interner Strukturen umzusetzen.*

Das hier beschriebene Modell kann in der Regel zur Beschreibung der funktionalen Abläufe der oben beschriebenen Aufgaben verwendet werden. Werden andere oder zusätzliche Aufgaben von einer Firewall verlangt, so muss auch das hier gegebene Modell entsprechend angepasst werden.

Komponenten, Systeme und Architekturen. Die von einer Firewall zu implementierenden Funktionen müssen technisch in einem Firewall-System, beschrieben durch die Firewall-Architektur und bestehend aus den einzelnen Firewall-Komponenten sowie der Festlegung ihrer Interaktionsprinzipien, realisiert werden.

> *Eine* ***Firewall-Komponente*** *stellt die technische Umsetzung von mindestens einer Teilmenge der von einer Firewall zu erfüllenden Aufgaben dar.*
>
> *Ein* ***Firewall-System*** *besteht aus mindestens einer Firewall-Komponente. Das Firewall-System stellt die technische Umsetzung der von einer Firewall zu erfüllenden Aufgaben dar.*
>
> *Eine* ***Firewall-Architektur*** *beschreibt den Aufbau einzelner Firewall-Komponenten, sowie das Zusammenspiel der einzelnen Firewall-Komponenten innerhalb des Firewall-Systems.*

Bei der Festlegung der grundlegenden Funktionen einer Firewall findet sich in der Literatur große Übereinstimmung. Hingegen ist die technische Umsetzung der Funktionen, die Beschreibung der

Firewall-Architektur, nicht allgemeingültig festgelegt. Demnach ergeben sich für die Optimierung eines Firewall-Systems für einen bestimmten Zweck Variablen hinsichtlich der technischen Umsetzung der Funktionen, nicht aber hinsichtlich der Funktionen selbst.

Prinzipiell ist jede beliebige Implementierung einer Firewall, die die festgelegten Firewall-Funktionen bereitstellt, zulässig. Dennoch gibt es einige bewährte Konzepte, die zur Implementierung von Firewall-Komponenten und Firewall-Systemen verwendet können, die im Folgenden ausführlich dargelegt werden.

2.2.2 Firewall-Komponenten

Zur Umsetzung der zuvor beschriebenen Firewall-Funktionen werden in der Regel die hier beschriebenen Standard-Firewall-Komponenten verwendet. Zur Darstellung der Funktionsweise wird die folgende Symbolik verwendet.

Abbildung 3: Symbole zur Beschreibung von Firewall-Komponenten

Paketfilter. Ein Paketfilter wird als Teil einer Netzwerkkomponente (z.B. einer Bridge oder einem Router) die zum Verbinden zweier Netzwerke verwendet wird, realisiert. Dadurch wird die Netzwerkkomponente um eine Firewall-Funktionalität erweitert. Der Paketfilter prüft anhand zuvor festgelegter Kriterien, ob die von der Netzwerkkomponente weiterzuleitenden Datenpakete eine Bedrohung darstellen. Stellt ein Datenpaket eine Bedrohung dar, wird es verworfen und nicht weitergeleitet. Die an bestimmten Positionen enthaltenen Werte innerhalb der Datenpakete werden dabei mit den zuvor festgelegten Regeln (bzw. der Bedingung der jeweiligen Regel) verglichen, um zu entscheiden, ob eine Bedrohung vorliegt oder nicht. Im wesentlichen werden die in den Header-Feldern der Vermittlungs- und Transportschicht enthaltenen Werte innerhalb der Datenpakete zur Analyse verwendet. Da jede Prüfung eines Paketes ohne Berücksichtigung der bereits analysierten Pakete durchgeführt wird, kann ein Paketfilter nur zustandslose Prüfungen durchführen. Für zustandsbehaftete Analysen müssen Stateful-Filter oder Proxies verwendet werden. Eine Prüfung applikationsspezifischer Parameter ist ebenfalls nicht möglich, da die Informationen der Applikationsschicht durch den Paketfilter nicht ausgewertet werden. IP-Paketfilter untersuchen beispielsweise die Pakete nach Typ des Pakets (z.B. TCP oder UDP), nach IP-Adressen des Senders und Empfängers, sowie nach den Port-Nummern des Senders und Empfängers. Pakete, die an einen bestimmten Empfänger gerichtet sind, können dann als Bedrohung erkannt werden - wenn dies in den Regeln so festgelegt wurde - und die Weiterleitung dieser Pakete an den Empfänger verhindert werden.

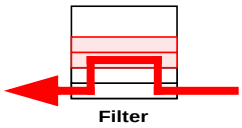


Abbildung 4: Paketfilter auf Router-Basis

Eine Firewall, die nur durch die Verwendung eines Paketfilters realisiert wird, setzt demnach die in Abschnitt 2.2.1 festgelegten Aufgaben einer Firewall folgendermaßen um:

- **Zugriffskontrolle:** Der Paketfilter kann die Identität der Kommunikationsteilnehmer anhand der verwendeten Port-Nummern und der IP-Adressen bestimmen.
- **Filtern und Modifizieren:** Die Firewall entscheidet, ob die Daten weitergeleitet werden oder nicht. Eine Modifikation der Daten ist vor dem Weiterleiten nicht möglich.
- **Verbergen:** Ein Verbergen der internen Netzstrukturen ist mit einem Filter nicht möglich.
- **Audit:** Wird ein Netzwerkpaket durch die Firewall verworfen und nicht weitergeleitet, so wird dies entsprechend vermerkt.

Stateful-Filter. Bei einem Stateful-Filter handelt es sich um einen Paketfilter, der unter anderem in der Lage ist, bei der Prüfung der Pakete auf Informationen zurückzugreifen, die durch die Prüfung von zuvor verarbeiteten Paketen gewonnen wurden. Ein Stateful-Filter kann in der Lage sein zu prüfen, ob ein TCP-Paket zu einer bereits ordnungsgemäß geöffneten TCP-Verbindung gehört oder nicht. Der Stateful-Filter “merkt” sich, dass zwischen dem Sender und Empfänger bereits TCP-Pakete für den TCP-Verbindungsaufbau in der richtigen Reihenfolge ausgetauscht wurden. Wird dann zu einem späteren Zeitpunkt ein TCP-Paket durch den Stateful-Filter geprüft, kann dieser feststellen, ob das Paket den festgelegten Filterregeln entspricht und zusätzlich in den Kontext der aktuellen TCP-Verbindungen passt. Ein Stateful-Filter kann auch in der Lage sein, einzelne Protokolle der Applikationsschicht zu “verstehen” und dadurch auch Informationen zur Prüfung der Daten heranziehen, die nur auf dieser Schicht verfügbar sind. Ein Stateful-Filter, der die Applikationssemantik des FTP-Protokolls “versteht” kann die auf dem FTP-Kontrollkanal ausgehandelten IP-Adressen und Port-Nummern für den FTP-Datenkanal ermitteln. Diese Informationen stehen dann für die Prüfung der folgenden TCP-Pakete zur Verfügung; TCP-Pakete können als Pakete einer FTP-Datenverbindung identifiziert werden.

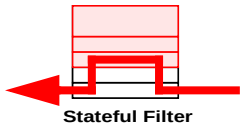


Abbildung 5: Stateful-Filter

Eine Firewall, die nur durch die Verwendung eines Stateful-Filter realisiert wird, kann demnach folgende Aufgaben erfüllen:

- **Zugriffskontrolle:** Der Stateful-Filter kann die Identität der Kommunikationsteilnehmer anhand der verwendeten Portnummern und der IP-Adressen bestimmen. Zusätzlich können benutzerbezogene Informationen, die innerhalb der Applikationsschicht eventuell vorhanden sind, verwendet werden.
- **Filtern und Modifizieren:** Die Firewall entscheidet, ob die Daten weitergeleitet werden oder nicht. Eine Modifikation der Daten ist vor dem Weiterleiten möglich.
- **Verbergen:** Ein Verbergen der internen Netzstrukturen ist mit einem Stateful-Filter allein nicht möglich.
- **Audit:** Werden Daten durch die Firewall verworfen oder modifiziert, so wird dies entsprechend vermerkt.

Proxy. Ein Proxy wird in den Kommunikationspfad zwischen zwei Kommunikationspartnern eingebracht, indem er für die jeweiligen Kommunikationsteilnehmer den Kommunikationsendpunkt widerspiegelt. Dadurch wird die Verbindung zwischen zwei Kommunikationspartnern durch den Proxy aufgetrennt. Alle Datenpakete, die an den Kommunikationsteilnehmer gesendet werden, werden durch den Proxy entgegengenommen und auf Applikationsebene verarbeitet. Dazu benötigt der Proxy "Wissen" über das der Kommunikation zugrunde liegende Applikationsprotokoll. Bei der Verarbeitung wird geprüft, ob sich durch die Daten Bedrohungen ergeben können. Dies geschieht anhand zuvor festgelegter Kriterien. Als Entscheidungsgrundlage, ob die Daten an den Empfänger weitergeleitet werden oder nicht, können nur die aus der Applikationsschicht gewonnenen Informationen verwendet werden. Informationen, die in den Daten der unteren Schichten enthalten sind, stehen einem Proxy nicht unmittelbar zur Verfügung.

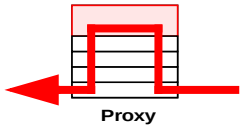


Abbildung 6: Proxy

Eine Firewall, die nur durch die Verwendung eines Proxies realisiert wird, kann demnach folgende Aufgaben erfüllen:

- **Zugriffskontrolle:** Der Proxy kann die Identität der Kommunikationsteilnehmer anhand der benutzerbezogenen Informationen, die innerhalb der Applikationsschicht eventuell vorhanden sind, prüfen.
- **Filtern und Modifizieren:** Die Firewall entscheidet, ob die Daten weitergeleitet werden oder nicht (Filterung). Eine Modifikation der Daten ist vor dem Weiterleiten möglich.
- **Verbergen:** Da die Kommunikationsteilnehmer durch den Proxy auf allen Schichten vollständig voneinander getrennt sind, können durch einen Proxy die Netzstrukturen verborgen werden.
- **Audit:** Werden Daten durch die Firewall verworfen oder modifiziert, so wird dies entsprechend vermerkt.

Network Address Translation (NAT). Eine NAT-Komponente ermöglicht es im Wesentlichen, die IP-Adressen der über die Komponente laufenden IP-Pakete zu modifizieren. Welche Adressen wie umgesetzt werden ist durch die Regeln der NAT-Komponente festgelegt. Wird für ein Paket eine Adressumsetzung durchgeführt, so muss auch für alle anderen Pakete des selben Datenstroms die gleiche Umsetzung durchgeführt werden, damit eine Kommunikation möglich ist. Neben dem Effekt, dass durch diesen Mechanismus ein gesamtes Subnetz auf eine IP-Adresse abgebildet werden kann [22], beinhaltet dieser Mechanismus auch eine Schutzfunktion. Rechner können über die NAT-Komponente hinweg nur dann adressiert werden, wenn innerhalb der Regeln eine entsprechende Adressumsetzung definiert ist.

Abbildung 7: NAT-Komponente

Eine Firewall, die nur durch die Verwendung einer NAT-Komponente realisiert wird, kann demnach folgende Aufgaben erfüllen:

- **Zugriffskontrolle:** Die NAT-Komponente kann die Identität der Kommunikationsteilnehmer anhand der verwendeten Port-Nummern und der IP-Adressen bestimmen.
- **Filtern und Modifizieren:** Die NAT-Komponente entscheidet, ob die Daten weitergeleitet werden oder nicht. Es wird entschieden ob eine Adressumsetzung für eine bestimmte Kommunikationsverbindung durchgeführt wird oder nicht. Eine Modifikation der Daten ist vor dem Weiterleiten nötig, um die verwendeten IP-Adressen und Port-Nummern zu verändern.
- **Verbergen:** Netzstrukturen werden durch die NAT-Komponente vollständig verborgen.
- **Audit:** Werden Daten durch die NAT-Komponente verworfen oder modifiziert, so wird dies entsprechend vermerkt.

2.2.3 Firewall-Systeme

Um alle für eine Firewall notwendigen Funktionen zu implementieren, werden in der Regel verschiedene der zuvor beschriebenen Firewall-Komponenten, die meist jeweils nur einen Teil der Funktionen bereitstellen, zu einem Firewall-System zusammengefasst.

Hybridsystem. Eine gängige Kombination besteht darin, einen Proxy mit einem Stateful-Filter und einer NAT-Komponente zu verbinden. Der Proxy (für verschiedene Applikationstypen) befindet sich dabei auf demselben Gerät, wie der Stateful-Filter und die NAT-Komponente. Die Komponenten sind dabei in der Lage, über eine geeignete Schnittstelle miteinander zu interagieren. Das Hybridsystem stellt damit die Funktionalität bereit, die sich durch Addition der Funktionalität der einzelnen Komponenten ergibt. Dadurch, dass eine Interaktionsmöglichkeit zwischen Proxy und Stateful-Filter besteht, kann der Stateful-Filter sich darauf beschränken, die Analyse der Daten auf Vermittlungs- und Transportschicht zu übernehmen. Die Analyse der Applikationsdaten wird durch den Proxy übernommen. Es ist in einer weiteren Variation ebenfalls möglich, auf den Proxy zu verzichten, indem seine Funktionen durch den Stateful-Filter vollständig erbracht werden. Es besteht demnach die Wahl, den Stateful-Filter entsprechend komplex zu gestalten und dafür auf den Proxy zu verzichten oder aber den Stateful-Filter einfach zu gestalten und einen Proxy bereitzustellen. Heutige auf dem Markt erhältliche Firewall-Systeme entsprechen in ihrem inneren Aufbau diesen Hybridsystemen. Beispiele dafür sind Checkpoints Firewall-1 [23] oder Ciscos PIX [24]. Hybridsysteme werden in der Praxis häufig als einziges Element verwendet, um eine Firewall aufzubauen.

Verteilte Systeme. Oft werden die zuvor beschriebenen Komponenten so an einer Netzgrenze verwendet, dass die einzelnen Komponenten von den Datenströmen nacheinander durchlaufen werden. Zwischen den einzelnen verwendeten Komponenten besteht dabei aber meist keine Mög-

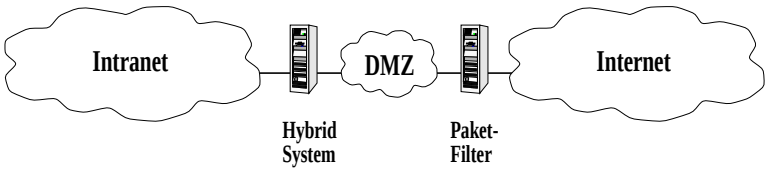


Abbildung 8: Verteiltes Firewall-System mit demilitarisierter Zone

lichkeit der Interaktion. Durch die Hintereinanderschaltung wird zum einen erreicht, dass alle für eine Firewall nötigen Funktionen erbracht werden. Zum anderen aber kann durch die Hintereinanderschaltung verschiedener Komponenten erreicht werden, dass ein möglicher Fehler in einer Komponente nicht zu einem Verlust der Schutzfunktion der gesamten Firewall führt. Eine häufig verwendete Hintereinanderschaltung von Komponenten ist in Abbildung 8 dargestellt. Der Paketfilter ist dem Hybridsystem vorgeschaltet. Sollte der Paketfilter fehlerhaft sein, so muss von einem Angreifer immer noch das Hybridsystem überwunden werden. Weitere Beschreibungen solcher verteilter Firewall-Systeme finden sich in [9], [10] und [21].

Parallele Systeme. Die herkömmlichen Firewall-Architekturen bestehen zumeist aus jeweils einer der oben beschriebenen Firewall-Komponenten. Jedoch reichen in Hochgeschwindigkeitsnetzen die von diesen Systemen bereitgestellten Übertragungsraten nicht mehr aus. Eine Möglichkeit, den hohen Datenraten gerecht zu werden, ist die parallele Verwendung mehrerer gleicher Firewall-Komponenten. Einfache Paketfilter mit gleicher Konfiguration können, wie in [25]

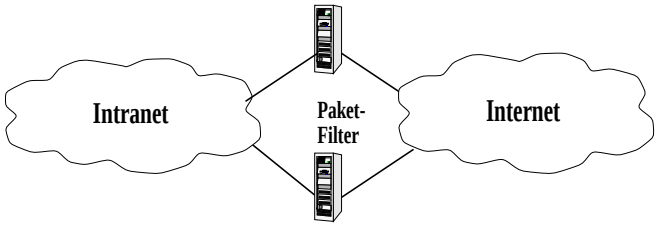


Abbildung 9: Parallele Paketfilter

beschrieben, parallel eingesetzt werden, um die mögliche Datenrate zu erhöhen. Mehrere Stateful-Filter sowie Proxies können so betrieben werden, dass eine Lastverteilung zwischen ihnen stattfindet. Diese Techniken können nicht mehr ohne Weiteres verwendet werden, wenn sich beispielsweise Konfigurationen dynamisch ändern oder aber nicht sichergestellt werden kann, dass alle Daten einer Kommunikationsverbindung über dieselben Komponenten laufen.

2.3 Multimedia-Applikationen

Der Begriff “Multimedia” bzw. “Multimedia-Applikation” wird mit verschiedensten Bedeutungen innerhalb der Literatur verwendet. Innerhalb dieser Arbeit wird das Themengebiet “Multimedia” entsprechend der in [26] gegebenen Festlegungen verstanden. Bezogen auf diese Definition liegt der Fokus dieser Arbeit auf Multimedia-Applikationen, die kontinuierliche Medien verarbeiten. Diese Medien werden dazu durch die Applikation über ein Netzwerk transportiert. Entsprechend dieser Einschränkung wird der Begriff “Multimedia-Applikation” innerhalb dieser Arbeit und als Basis für die nachfolgenden Definitionen verwendet.

2.3.1 Definitionen

Um die Begriffe “Multimedia-Applikation” und “Multimedia-Protokoll” erläutern zu können müssen zunächst die Begriffe “Kanal” und “Sitzung” erläutert werden. Beide Begriffe können dazu verwendet werden, den von einer Multimedia-Applikation verarbeiteten Datenfluss auf verschiedenen Granularitätsstufen zu beschreiben.

*Ein **Strom (Flow)** wird als ein Fluss semantisch zusammenhängender Datenpakete verstanden, der in TCP/IP-Netzen durch ein 5-Tupel (Quelladresse, Quellport, Zieladresse, Zielport und Transportprotokollnummer) beschrieben ist. In vielen Dokumenten wird der Begriff **Kanal (Channel)** verwendet; auch in der vorliegenden Arbeit werden beide Begriffe synonym verwendet.*

*Der Begriff **Sitzung (Session)** ist angelehnt an zeitlich abgeschlossene und inhaltlich zusammenhängende Aktivitäten. Als eine solche Aktivität kann z.B. ein Gespräch oder das Ansehen eines Films verstanden werden. Eine Sitzung beinhaltet einen oder mehrere Kanäle, welche die für die Durchführung der Aktivität notwendigen Daten transportieren.*

*Ein **Multimedia-Protokoll** definiert unter anderem die Struktur einer Sitzung. Das Multimedia-Protokoll definiert, wieviele Ströme für eine Sitzung verwendet werden, wie die einzelnen Ströme initiiert und verwendet werden, sowie welche Inhalte und in welcher Form diese in den Strömen transportiert werden.*

*Eine **Multimedia-Applikation** wird verwendet, um über ein Netzwerk kontinuierliche und diskrete Medien zu transportieren und diese dann zu verarbeiten. Zusätzlich kann eine Multimedia-Applikation diskrete Medien transportieren und verarbeiten. Der Transport der Medien erfolgt mit Hilfe der innerhalb des entsprechenden Multimedia-Protokolls festgelegten Sitzung. Außerdem kann die Multimedia-Applikation zur Erbringung ihrer Funktionen zusätzliche Protokolle verwenden.*

2.3.2 Kommunikationsablauf

Der Fokus dieser Arbeit liegt auf Multimedia-Protokollen, die mindestens zwei Kanäle pro Sitzung verwenden. Ein Kanal wird für den Transport von Signalisierungsdaten verwendet, der andere zum Transport eines kontinuierlichen Mediums. Diese Multimedia-Applikationen kommunizieren in der Regel auf die im Folgenden beschriebene Art und Weise.

Abbildung 10: Kommunikationsablauf einer Multimedia-Applikation

Zwischen beiden Endpunkten (EP1 und EP2) wird ein Kontrollkanal zur Steuerung einer Sitzung initiiert. Nachdem der Kontrollkanal aufgebaut ist, wird ein weiterer Kanal, über den das kontinuierliche Medium (z.B. Audio oder Video) übertragen wird, etabliert. Die für diesen Medienkanal zu verwendenden Parameter - wie z.B. das verwendete Protokoll, die zu verwendenden Protokollparameter, die verwendeten Kodierungsverfahren, usw. - werden dabei über den Kontrollkanal zwischen den Kommunikationspartnern ausgehandelt. Das Ende einer Sitzung wird zwischen den Endpunkten über den Kontrollkanal signalisiert, wenn nicht ein Timeout für den Abbau einer Sitzung verwendet wird. Alle bestehenden Kanäle werden dann abgebaut.

Kapitel 3: Multimedia-Applikationen und Firewalls

In diesem Kapitel wird beschrieben, welche Probleme sich bei der Integration von Multimedia-Applikationen in durch Firewalls geschützte Netzwerke ergeben. Des Weiteren werden die Problemursachen beschrieben und eine Klassifikation derselben gegeben. Aus dieser Klassifikation werden Lösungsansätze abgeleitet, aus denen sich dann sowohl Anforderungen für Firewalls als auch Anforderungen für Multimedia-Applikationen ergeben. Abschließend wird für verschiedene Multimedia-Applikationen (H.323-, SIP- und RTSP-Applikationen) exemplarisch aufgezeigt, dass die gegebene Klassifikation und damit die aus ihnen abgeleiteten Anforderungen in konkreten Anwendungsszenarien Gültigkeit besitzen.

3.1 Auftretende Probleme

Sollen Multimedia-Applikationen in einer Umgebung eingesetzt werden, in der auch konventionelle Firewalls verwendet werden, so führt dies zu einer Beeinträchtigung der Funktionalität der Multimedia-Applikation, und/oder zu einer Beeinträchtigung der Funktionalität der Firewall. So können sich im Wesentlichen folgende funktionale Beeinträchtigungen bzw. Probleme ergeben:

Verhinderung der Nutzung einer Multimedia-Applikation. Die zur Erfüllung der von einer Multimedia-Applikation zu erbringenden Aufgaben notwendigen Kommunikationskanäle können nicht über die Firewall hinweg etabliert werden. In diesem Fall kann die Multimedia-Applikation nicht genutzt werden, sowie sich die beteiligten Kommunikationsendpunkte auf verschiedenen Seiten der Firewall befinden. Diese vollständige funktionale Beeinträchtigung der Applikation kann eintreten, wenn die Firewall nicht in der Lage ist, alle zu einer Sitzung gehörenden Datenströme zu identifizieren und deshalb bestimmte Datenströme nicht passieren lässt.

Beeinträchtigung der Funktionalität einer Multimedia-Applikation. Die Multimedia-Applikation kann nur mit Einschränkungen verwendet werden. Bestimmte funktionale Eigenschaften der Applikation stehen zwar zur Verfügung, nicht aber in dem Maße, wie dies benötigt wird. Zum Beispiel wird ein Audiostrom zwar von einem Sender zum Empfänger transportiert, die Firewall verzögert aber die Datenpakete in solch einem Maß, dass die Audiodaten nicht mehr sinnvoll wiedergegeben werden können.

Beeinträchtigung der Schutzfunktion der Firewall. Die Firewall muss mit Einschränkungen ihrer Schutzfunktion betrieben werden. Die Multimedia-Applikation kann dann ohne die zuvor beschriebenen Einschränkungen verwendet werden, aber es muss auf einen Teil der Schutzfunktionen, die die Firewall bereitstellt, verzichtet werden. Ist die Firewall beispielsweise nicht in der

Lage, alle zu einer Sitzung gehörenden Datenströme zu identifizieren, kann die Firewall statisch so eingestellt werden, dass sie alle potentiell auftretenden Datenströme passieren lässt. Dies ermöglicht einem Angreifer, diese dauerhaft geöffneten Kommunikationswege durch die Firewall für einen Angriff zu verwenden.

In der Literatur (z.B. [27], [28], [29]) finden sich zahlreiche Beschreibungen von Problemen, die bei der Verwendung von Multimedia-Applikationen in Firewall geschützten Netzen auftreten. Diese beschriebenen Problemfälle lassen sich in die oben gegebenen Kategorien einordnen.

3.2 Ursachen der Probleme

Für ein fundiertes Problemverständnis und als Basis für eine Evaluierung bestehender Vorschläge sowie die Entwicklung geeigneter Lösungsansätze ist eine Strukturierung der Probleme hinsichtlich ihrer Ursachen notwendig. In den folgenden Abschnitten werden die Ursachen dargestellt und strukturiert.

3.2.1 Beschreibung

Die zuvor beschriebenen Probleme können hinsichtlich der Suche nach den Ursachen von zwei verschiedenen Standpunkten aus betrachtet werden. Der erste Standpunkt rückt die Firewall als Problemverursacher in den Mittelpunkt. Dieser Standpunkt wird in der Regel durch Hersteller und Nutzer der Multimedia-Applikationen eingenommen. Der zweite Standpunkt hat die Multimedia-Applikation als Problemverursacher im Fokus und wird oft durch Firewall-Hersteller bzw. Firewall-Administratoren vertreten. Je nach vertretenem Standpunkt kann die Ursache der auftretenden Beeinträchtigungen folgendermaßen verstanden werden:

> ***Firewallzentrierte Sicht:*** *Heute verwendete Firewalls sowie die ihnen zugrundeliegenden Techniken wurden entwickelt, um mit herkömmlichen Applikationen zusammenzuarbeiten. Multimedia-Applikationen standen beim Entwurf dieser Systeme, bzw. der zugrunde liegenden Techniken, nicht im Vordergrund. Dementsprechend sind diese nicht für den Betrieb im Zusammenhang mit Multimedia-Applikationen ausgelegt oder optimiert.*
>
> ***Applikationszentrierte Sicht:*** *Multimedia-Applikationen bzw. die zugrunde liegenden Protokolle werden in der Regel entworfen, ohne zu berücksichtigen, dass diese in einem Umfeld betrieben werden müssen, in dem Firewalls verwendet werden. Dementsprechend treten beim Betrieb der Multimedia-Applikationen in einer Umgebung, in der Firewalls verwendet werden, die oben beschriebenen Probleme auf.*

Beiden Betrachtungsweisen liegt zugrunde, dass Multimedia-Applikationen sich von “traditionellen Applikationen” (z.B. Http, Telnet) hinsichtlich verschiedener Charakteristika unterscheiden. Aus Sicht der Firewall fehlen Techniken, um die durch diese Charakteristika verursachten Beeinträchtigungen auflösen zu können. Aus Sicht der Multimedia-Applikation entstehen die Beeinträchtigungen durch das Vorhandensein eben dieser Charakteristika.

Es zeigt sich, dass die Betrachtung dieser Charakteristika für den Entwurf geeigneter Lösungskonzepte wesentlich ist. Eine Klassifizierung und Beschreibung der Charakteristika wird im Folgenden gegeben.

3.2.2 Klassifizierung

Multimedia-Applikationen unterscheiden sich hinsichtlich vieler spezifischer Eigenschaften signifikant von “traditionellen Applikationen”. Die im Folgenden aufgeführten Charakteristika, die im Zusammenhang mit Firewalls wesentlich für die bestehenden Probleme verantwortlich sind, können in drei Gruppen unterteilt werden. Die Zuordnung eines Charakteristikums zu einer Gruppe geschieht aus drei Gründen.

- Die Charakteristika sind thematisch geordnet. Es werden die Themengebiete Protokolle, Applikationen und Leistung betrachtet.
- Die Charakteristika in einer Gruppe sind zum Teil interdependent. Die Änderung einer Eigenschaftsausprägung kann Veränderungen der Ausprägung der anderen Eigenschaften nach sich ziehen.
- Aufgrund der teilweisen Abhängigkeit der Charakteristika adressieren Problemlösungen für die durch eine Eigenschaft verursachten Probleme zumeist alle Eigenschaften einer Gruppe.

Wie die einzelnen Charakteristika genau Probleme verursachen und dass die einzelnen Charakteristika in konkreten Szenarien angetroffen werden, wird anhand verschiedener Beispiele nachfolgend (siehe Abschnitt 3.4 und Abschnitt 3.5) gezeigt.

Protokollcharakteristika. Die erste Gruppe fasst diejenigen Charakteristika zusammen, die auf Protokollebene angesiedelt sind. Alle Multimedia-Applikationen, die auf dem selben Protokoll (z.B. H.323, SIP, RTSP) aufbauen besitzen diese Charakteristika in gleicher Ausprägung.

> ***Komplexität der Protokolle:*** *Die von Multimedia-Applikationen verwendeten Multimedia-Protokolle besitzen in der Regel eine deutlich höhere Komplexität als traditionelle Applikationen. Dies gründet im Wesentlichen darauf, dass für die Behandlung der verschiedenen, dynamisch ausgehandelten Datenströme einer Sitzung zum einen verschiedene Teilprotokolle verwendet werden, zum anderen die Verwaltung der verschiedenen Ströme eine vergleichsweise komplexe Protokollmaschine benötigt.*
>
> ***Mehrere Ströme in einer Sitzung:*** *Eine Multimedia-Sitzung besteht aus mehreren einzelnen Datenströmen, die für verschiedene Aufgaben verwendet werden. Dies unterscheidet eine Multimedia-Sitzung von der Sitzung einer traditionellen Applikation, die in der Regel nur einen Strom pro Sitzung verwendet.*
>
> ***Dynamisches Verhalten:*** *Ein Multimedia-Protokoll zeichnet sich durch ein sehr dynamisches Verhalten aus. Viele der verwendeten Ströme benutzen dynamisch zwischen Sender und Empfänger ausgehandelte Ports. Auch die verwendete Bandbreite einzelner Ströme kann sich während der Dauer einer Sitzung dynamisch ändern. Ebenso kann die Anzahl der verwendeten Ströme während der Dauer einer Sitzung geändert werden.*

Applikationscharakteristika. Die in dieser Gruppe aufgeführten Charakteristika sind hauptsächlich der Applikationsebene zuzuordnen. Verschiedene Applikationen, die auf dem gleichen Protokoll aufbauen, prägen die hier gelisteten Charakteristika verschieden aus.

Szenarienvielfalt und Szenarienkomplexität: *Im Vergleich zu traditionellen Applikationen verwenden Multimedia-Applikationen in der Regel eine komplexe Infrastruktur. Das Design bzw. der Umfang der für den jeweiligen Anwendungsfall verwendeten Infrastruktur führt zur Ausprägung verschiedener Szenarien. Den verschiedenen möglichen Szenarien liegen meist unterschiedliche Routing-Mechanismen der Signalisierungs- bzw. Medienströme zugrunde.*

Interpretationsmöglichkeiten der Protokollstandards: *Verschiedene Multimedia-Applikationen verwenden die Multimedia-Protokolle meist auf verschiedene Arten. Dies liegt daran, dass die sehr umfangreichen Standards, die die Multimedia-Protokolle beschreiben, an manchen Stellen Interpretationsmöglichkeiten zulassen. Dadurch entstehen Multimedia-Applikationen, die dasselbe Multimedia-Protokoll verwenden, aber ein sehr unterschiedliches Kommunikationsverhalten aufweisen.*

Verwendung zusätzlicher Protokolle: *Multimedia-Applikationen können zusätzliche Protokolle verwenden, um bestimmte Aufgaben zu erfüllen, die nicht durch das verwendete Multimedia-Protokoll abgedeckt sind. Beispielsweise können bestimmte Informationen, die zur Initiierung einer Multimedia-Sitzung nötig sind, in einer Datenbank abgelegt werden. Die Abfrage der Datenbank erfolgt über ein Protokoll, das nichts mit dem Multimedia-Protokoll zu tun hat. Dennoch gehört die Abfrage der Datenbank zu der Multimedia-Sitzung.*

Leistungscharakteristika. In dieser Gruppe zusammengefasste Charakteristika beschreiben die für den Betrieb einer Multimedia-Applikation notwendigen Leistungsanforderungen.

Hohe Datenrate: *Multimedia-Applikationen verwenden Ströme, die eine sehr hohe Bandbreite - teilweise über einen sehr langen Zeitraum - benötigen. Insbesondere Videoübertragungen benötigen eine hohe Bandbreite.*

Dienstgüteanforderungen: *Zusätzlich unterliegen die übertragenen Daten bestimmten Dienstgüteanforderungen. Diese Dienstgüteanforderungen müssen eingehalten werden, damit die transportierten Inhalte verwendet werden können.*

3.3 Lösungsansätze

Die im vorhergehenden Abschnitt beschriebenen Ursachen der Probleme führen zunächst zu zwei generellen Optimierungsmöglichkeiten. Diese können genutzt werden, um Multimedia-Applikationen und Firewalls für einen gemeinsamen, beeinträchtigungsfreien Betrieb auszulegen.

Optimierungsmöglichkeiten aus Sicht der Firewall: *Firewalls müssen so ausgelegt werden, dass sie die Charakteristika einer Multimedia-Applikation unterstützen können. Es existieren bewährte Firewall-Techniken, um klassische Applikationen sinnvoll zu handhaben. Entsprechend müssen Methoden gefunden werden, um die Charakteristika von Multimedia-Applikationen ebenso unterstützen zu können. Bei der Auslegung der Firewall ist zu beachten, dass die in Abschnitt 2.2.1 beschriebenen Firewall-Funktionen erhalten bleiben. Die Firewall-Architektur kann unter Beibehaltung der Firewall-Funktionen verändert werden.*

Optimierungsmöglichkeiten aus Sicht der Multimedia-Applikation: *Die Multimedia-Applikationen sowie die zugrunde liegenden Multimedia-Protokolle müssen dahingehend ausgelegt werden, das diese Firewalls unterstützen. Dies bedeutet einerseits, dass Multimedia-Applikationen, die für eine Firewall problematischen Charakteristika möglichst wenig*

ausprägen sollte. Andererseits sollte eine Multimedia-Applikation die Existenz von Firewalls hinsichtlich ihres Designs vorsehen.

Es fällt auf, das die zweite Möglichkeit aus zwei Gründen nur bedingt nutzbar ist. Erstens kann eine Multimedia-Applikation nicht sinnvoll entworfen werden, wenn die spezifischen Charakteristika nicht verwendet werden. Es ist nicht möglich, eine Multimedia-Applikation zu entwerfen, die die oben beschriebenen Dienstgüteanforderungen nicht besitzt. Es bleibt einzig die Möglichkeit, eine geringe Ausprägung der jeweiligen Eigenschaften anzustreben. Auch ist diese Optimierungsmöglichkeit in der Praxis kaum nutzbar, da eine Neudefinition schon bestehender Protokolle bzw. der Neuentwurf vieler bestehender Applikationen nicht durchführbar ist.

Zunächst können beide Optimierungsmöglichkeiten getrennt voneinander verwendet werden. Bei einer Kombination beider Optimierungsmöglichkeiten ergibt sich allerdings ein größerer Lösungsraum als bei der Verwendung von nur einer Optimierungsmöglichkeit. Dadurch kann möglicherweise eine für das Problem bessere Lösung gefunden werden als bei der Berücksichtigung nur einer Möglichkeit. Zusätzlich beeinflussen sich beide Möglichkeiten gegenseitig, weshalb ebenfalls eine vollständig getrennte Betrachtung beider Möglichkeiten nur im ersten Ansatz verwendet werden kann. Wird die Multimedia-Applikation geändert, um eine Firewall explizit in die Kommunikation einzubinden, hat dies Auswirkungen auf die Designmöglichkeiten der verwendeten Firewall-Architektur.

Aus den obigen Betrachtungen ergibt sich nun der folgende, generelle Lösungsansatz:

Optimierungsmöglichkeit: *Die Firewall-Architektur muss so angepasst werden, dass sie die Charakteristika einer Multimedia-Applikation vollständig unterstützt. Soweit möglich, soll die Multimedia-Applikation ebenfalls verändert werden, damit das Anpassen der Firewall-Architektur leicht vorgenommen werden kann.*

Bei der Optimierung des Zusammenspiels zwischen Firewall und Multimedia-Applikation muss die Sicherheit beachtet werden. Diese Randbedingung wird nachfolgend genauer betrachtet.

3.3.1 Sicherheitsaspekte

Wie zuvor beschrieben, liegt die wesentliche Optimierungsmöglichkeit in der Anpassung der verwendeten Firewall-Architektur. Bei der Festlegung bzw. Veränderung einer Firewall-Architektur ist zu beachten, dass die von der Firewall zu erbringenden Schutzfunktionen (siehe Abschnitt 2.2.1) erhalten bleiben. Ebenso ist zu beachten, dass bei einer Veränderung der Multimedia-Applikation die von ihr erbrachten Schutzfunktionen (z.B. Vertraulichkeit durch Ende-zu-Ende Verschlüsselung der Medienströme) erhalten bleiben. Die in der vorliegenden Arbeit im Vordergrund stehenden Schutzfunktionen sind solche, die durch eine Firewall umgesetzt werden. Aus diesem Grund soll auf diese im Folgenden genauer eingegangen werden.

Im Gegensatz zu traditionellen Applikationen, bei denen die in einer Firewall umgesetzten Sicherheitsfunktionen an den einen der Applikation zugrunde liegenden Strom gebunden sind, müssen bei Multimedia-Applikationen verschiedene Stromtypen in Betracht gezogen werden. Die von einer Multimedia-Applikation verwendeten Ströme können durch eine Firewall hinsichtlich der Umsetzung von Schutzfunktionen in einer Firewall getrennt betrachtet werden.

Signalisierungsstrom. Die von einer Multimedia-Applikation verwendeten Signalisierungsströme müssen von einer Firewall auf Netzwerk, Transport und Applikationsebene überprüft werden.

Bedrohungen können entstehen, indem durch einen Angreifer die Signalisierung einer Multimedia-Applikation manipuliert wird. Die Signalisierungsinformationen lösen innerhalb der von diesen Informationen durchlaufenen Protokollmaschinen (in Endsystemen sowie Infrastrukturkomponenten) Zustandsänderungen aus. Eine geeignete Manipulation der Signalisierungsinformationen kann daher durch einen Angreifer genutzt werden, um Multimedia-Applikationen sowie die von ihnen verwendete Infrastruktur in einen für einen Angreifer günstigen Zustand zu versetzen. Aus diesem Grund sind Signalisierungsinformationen innerhalb einer Firewall vor dem Weiterleiten zu überprüfen und gegebenenfalls zu verwerfen oder zu modifizieren.

Medienstrom. Die von einer Multimedia-Applikation verwendeten Medienströme müssen von einer Firewall auf Netzwerk- und Transportebene überprüft werden.

Die Firewall muss feststellen, ob ein auftretender Medienstrom im Kontext des aktuell ablaufenden Multimedia-Protokolls gültig ist. Des Weiteren muss die Firewall prüfen, ob die vereinbarte Stromspezifikation (5-Tupel) eingehalten wird. Auf eine Sicherheitsprüfung der durch die Medienströme transportierten Inhalte durch die Firewall kann in der Regel verzichtet werden. Zum einen kann eine Firewall nicht prüfen, ob die Inhalte (z.B. die gesprochenen Worte innerhalb eines RTP-Audioströme) eine Gefährdung darstellen. Zum anderen kann durch Manipulation der für diesen Strom verwendeten Protokoll-Elemente (z.B. Veränderung der RTP-Header eines Audiostroms) in der Regel keine für einen Angreifer nützliche Aktion ausgelöst werden.

Zur Erbringung der Sicherheitsfunktionen können demnach je nach Stromtyp unterschiedliche Ebenen verwendet werden. Dass diese Betrachtung verwendet werden kann, wird in Abschnitt 3.4.2 und Abschnitt 3.5 durch entsprechende Fall-Studien unterlegt.

3.3.2 Anforderungen an eine Multimedia-Firewall

Beachtet man die zuvor vorgestellten Optimierungsmöglichkeiten, sowie die problematischen Charakteristika der Multimedia-Applikationen und die einzuhaltenden Randbedingungen bezüglich der Sicherheit, kann folgendes Designprinzip für Firewall-Architekturen formuliert werden (siehe [30], [31] und [32]):

> ***Trennung von Signalisierungs- und Medienpfad:*** *Signalisierungsströme sowie Medienströme sollten in einer Multimedia-Firewall getrennt voneinander behandelt werden.*

Es ist möglich, die Signalisierungsströme sowie Medienströme auf getrennten Wegen durch ein Firewall-System zu leiten. Dies ermöglicht es, beide Wege an die jeweiligen speziellen Eigenschaften der Ströme anzupassen. Dies schließt insbesondere eine Anpassung an die verschiedenen umzusetzenden Sicherheitsfunktionen (siehe oben) ein. Eine Firewall-Architektur, die dieses Prinzip verwendet, kann deshalb besser den Leistungsanforderungen einer Multimedia-Applikation gerecht werden. Zusätzlich kann auch eine getrennte Dimensionierung beider Wege erfolgen, wodurch eine effiziente Nutzung von Ressourcen möglich ist.

Eine Trennung beider Pfade kann nur bis zu einem gewissen Grad erfolgen. Da beide Pfade zusammen eine logische Einheit (Sitzung) bilden, muss innerhalb einer Firewall eine zusammenhängende Sicht existieren. Dazu ist der Austausch von Informationen zwischen die den Signalisierungspfad verarbeitenden Elementen und die den Medienpfad verarbeitenden Elementen notwendig. Dieser Informationsaustausch ist nötig, damit die Firewall mit den Protokollcharakteristika *Mehrere Ströme in einer Sitzung* und *Dynamisches Verhalten* umgehen kann.

Werden Signalisierungspfad und Medienpfad getrennt betrachtet ergibt es sich, dass durch diese Modularisierung dann auch besser mit der Protokollcharakteristik *Komplexität der Protokolle* umgegangen werden kann.

Durch die Trennung können die Elemente für die Verarbeitung eines Pfades für jeden Pfad einzeln angepasst oder ausgetauscht werden. Dies ermöglicht es, eine Firewall an spezielle Applikationscharakteristika (was sich auf die Verarbeitungslogik des Signalisierungspfades auswirkt) anzupassen, ohne die Verarbeitungslogik des Medienpfades zu verändern. Zusätzlich kann die Verarbeitungslogik eines Pfades beibehalten werden, wenn ein weiteres, durch die Firewall zu unterstützendes Protokoll, sich nur hinsichtlich eines Pfades von einem bereits unterstützten Protokoll unterscheidet.

Die meisten der heute eingesetzten Firewalls setzen das hier dargestellte Designprinzip nicht um (siehe Kapitel 4), weshalb die in Abschnitt 3.1 beschriebenen Auswirkungen bei heute verwendeten Firewalls beobachtet werden können. Dass die Verwendung des hier beschriebenen Design-Prinzips die beschriebenen Vorteile, insbesondere die bessere Erfüllung der Leistungsanforderungen, mit sich bringt, wird in Kapitel 7 gezeigt. In Kapitel 5 und Kapitel 6 wird untersucht, welche Bausteine für die Realisierung des Designprinzips notwendig sind, und wie die Verarbeitungslogik der einzelnen Pfade realisiert werden kann.

3.3.3 Anforderungen an eine Multimedia-Applikation

Es lassen sich ebenfalls Designprinzipen für Multimedia-Applikationen entwickeln, so dass bei einer Einhaltung dieser Prinzipien die Multimedia-Applikation einfacher in ein Firewall-Umfeld integriert werden kann.

Es existieren verschiedene, nicht auf Multimedia-Applikationen im speziellen zugeschnittene Anforderungskataloge (z.B. generell für NAT [33], [34]; für das CORBA/RMI-Umfeld [35], [36]), die Design-Hinweise für Applikationen enthalten. Die meisten der dort beschriebenen Vorschläge konzentrieren sich auf die Unterdrückung der in Abschnitt 3.2.2 beschriebenen Charakteristika. Beispielsweise wird in [33] vorgeschlagen, keine IP-Adressen innerhalb der Applikationsschicht auszutauschen bzw. auszuhandeln. Dies verhindert das dynamische Aushandeln von Medienströmen hinsichtlich der Zieladressen. So werden die durch die Eigenschaft *Dynamisches Verhalten* verursachten Probleme abgemildert, die Möglichkeiten der Multimedia-Applikation aber auch eingeschränkt.

Eine alternative Möglichkeit besteht darin, die Multimedia-Applikationen dahingehend zu ändern, dass bestimmte, die problematischen Eigenschaften betreffende Informationen (z.B. dynamisch ausgehandelte Strombeschreibungen), an die Firewall übermittelt werden. Die

Firewall kann dann entsprechend leichter mit diesen Eigenschaften umgehen. Wenn die Informationen nicht über das Multimedia-Protokoll selbst (In-Band), sondern über eine zusätzliche Signalisierung an die Firewall übermittelt werden, kann dies über eine Erweiterung der Multimedia-Applikation, im Gegensatz zu einer Änderung des Multimedia-Protokolls, erfolgen.

Es können dementsprechend zwei Designprinzipien gegeben werden, wobei insbesondere das zuerst genannte nur unter bestimmten Umständen verwendet werden kann.

Unterdrücken der problematischen Charakteristika: *Eine Multimedia-Applikation kann so entworfen werden, dass die spezifischen, problemverursachenden Charakteristika nicht verwendet werden.*

Aktive Unterstützung der Firewall: *Die Multimedia-Applikation kann bestimmte die problematischen Eigenschaften betreffende Informationen an die Firewall übermitteln, um der Firewall den Umgang mit diesen Eigenschaften zu erleichtern. Die Übermittlung kann dabei In-Band oder durch externe Signalisierung erfolgen.*

Wie diese Designprinzipien (insbesondere das Zweite) umgesetzt werden können wird in Kapitel 4 und Kapitel 5 beschrieben.

3.4 H.323-Applikationen

IP-Telefonieapplikationen werden verwendet, um eine Audioverbindung zwischen zwei Endsystemen aufzubauen. Dabei wird, anders als bei der klassischen Telefonie, ein paketvermitteltes IP-Netzwerk als Trägermedium der Sprach- und Signalisierungsdaten verwendet. Die Applikationen können auf verschiedenen Protokollfamilien basieren, wobei im Wesentlichen die H.323- und die SIP-Protokollfamilie (siehe Abschnitt 3.5) eingesetzt wird.

In dieser Arbeit wird das H.323-Protokoll als durchgängiges Beispiel eines Multimedia-Protokolls verwendet. Deshalb wird im Folgenden das H.323-Protokoll kurz erläutert; eine vollständige Beschreibung des Protokolls kann in dieser Arbeit nicht gegeben werden, weshalb an den entsprechenden Stellen auf weiterführende Literatur verwiesen wird. Danach werden experimentell durchgeführte Angriffe auf H.323-Applikationen beschrieben, die die in Abschnitt 3.3.1 getroffenen Annahmen stützen. Anschließend wird dargestellt, in welcher Ausprägung die in Abschnitt 3.2.2 klassifizierten Charakteristika innerhalb von H.323-Applikationen zu finden sind.

3.4.1 H.323-Grundlagen

Im Jahr 1996 wurde von der ITU-T der Standard H.323 mit dem Ziel verabschiedet, multimediale Kommunikation über lokale Netzwerke, die keine Dienstgüte gewährleisten, zu ermöglichen. Die Kommunikationsdienste beinhalten Übertragungen von Audio, Video (optional) und Daten (optional) für Punkt-zu-Punkt und Multipoint-Konferenzen. Der H.323-Standard wird laufend weiter entwickelt; die einzelnen Entwicklungsstufen äußern sich in den verschiedenen Versionen des Standards. Aktuell ist die Version 4 des H.323-Standards [37].

Der H.323-Standard beschreibt die zur Multimedialen Kommunikation notwendigen Komponenten, die Kommunikation der Komponenten (Signalisierung und Medientransport), sowie zu verwendende Medienkodierungsverfahren. Innerhalb des H.323-Standards werden auch andere Protokolle wie das von der IETF standardisierte RTP, weitere ITU-Protokolle wie H.225.0, H.245 und Kodierungsprotokolle einbezogen, weshalb H.323 auch als Protokollfamilie bezeichnet wird. Außerhalb des Rahmens von H.323 sind Definitionen von Netzwerkschnittstellen, physischen Netzwerken oder verwendeten Netzwerkprotokollen. Außerhalb des Rahmens befindet sich dementsprechend auch die Definition der Interaktion mit Firewall-Systemen.

Signalisierung - RAS: H.225.0 [38]. Das Registration, Admission, Status (RAS) Protokoll wird für die Kommunikation zwischen einem Gatekeeper und anderen H.323-Komponenten verwendet. RAS wird beispielsweise verwendet, um Teilnehmer innerhalb eines Szenarios zu registrieren. Der Transport von RAS-Nachrichten erfolgt über UDP.

Signalisierung - Call Signalling: H.225.0 [38], Q.931 [39]. H.225.0 spezifiziert den Verbindungsaufbau zwischen zwei H.323-Endpunkten durch den Austausch von Q.931-Nachrichten. Zum Verbindungsaufbau gehört auch die Etablierung des Call Control Kanals. In Szenarien ohne Gatekeeper werden die Q.931-Nachrichten unmittelbar zwischen den Endpunkten ausgetauscht. In Szenarien mit Gatekeepern hingegen kann der Austausch der Nachrichten über die Gatekeeper erfolgen. Der Transport von Q.931-Nachrichten erfolgt über TCP.

Signalisierung - Call Control: H.245 [40]. Über den H.245-Kontrollkanal werden Kontrollnachrichten ausgetauscht, um die Arbeitsweise von H.323-Einheiten zu steuern. Dazu gehört beispielsweise der Austausch der technischen Fähigkeiten (z.B. die möglichen Medienkodierungsverfahren) und das Öffnen und Schließen der logischer Kanäle (z.B. der Audiokanäle). In Szenarien ohne Gatekeeper werden die H.245-Nachrichten unmittelbar zwischen den Endpunkten ausgetauscht. In Szenarien mit Gatekeepern hingegen kann der Austausch der Nachrichten über die Gatekeeper erfolgen. Der Transport von H.245-Nachrichten erfolgt über TCP.

Medien - Media/Media Control: RTP, RTCP[41]. Das Realtime Transport Protocol (RTP) wird verwendet, um kontinuierliche Medien (z. B. Audio oder Video) zu transportieren. Zusätzliche Kontrollinformationen während des Transports liefert das RTP Control Protocol (RTCP). Damit die kontinuierlichen Medien innerhalb des Empfängers verarbeitet werden können, werden über RTP nicht nur die Mediendaten, sondern auch zusätzliche Informationen (z.B. Zeitmarkierung, Sequenznumerierung) übermittelt. Während des RTP-Datentransports werden zwischen Sender und Empfänger RTCP-Nachrichten ausgetauscht. Diese liefern Informationen über auftretende Paketverzögerungen und Paketverluste beim Datentransport. Die Übermittlung der RTP und RTCP Nachrichten erfolgt über UDP.

Medien - Audiokodierung. Alle H.323-Terminals müssen mindestens ein Audiokodierungsverfahren unterstützen. Durch das Verfahren werden Audiosignale eines Mikrofons kodiert und umgekehrt eingehende kodierte Audiosignale für die Ausgabe am Lautsprecher dekodiert. Für die Kodierung und Dekodierung gibt es verschiedene ITU-Standards. Gemäß H.323 müssen Terminals den Standard G.711 unterstützen; optional können weitere Kodierungsverfahren verwendet werden. Der verwendete Algorithmus, sowie die notwendigen Parameter, werden während des Austauschs der Fähigkeiten unter H.245 ermittelt.

Medien - Videokodierung. Die Unterstützung eines Videokodierungsverfahrens in einem H.323-Terminal ist optional. Falls jedoch H.323-Terminals Videokommunikation unterstützen, sollte die Kodierung und Dekodierung von Video gemäß H.261 CIF implementiert sein. Optional können zusätzlich andere Verfahren verwendet werden. Die Bitrate, das Bildformat sowie Algorithmusoptionen werden während des Austauschs der technischen Fähigkeiten unter H.245 festgelegt.

Medien - T.120. Der Standard T.120 ist die Basis für die Austauschbarkeit von Daten zwischen einem H.323-Terminal und anderen H.323, H.324, H.320 oder H.310-Terminals. Es können ein oder mehrere Datenkanäle für den Austausch von Daten optional geöffnet werden. Entsprechend der jeweils verwendeten Datenanwendung kann der Datenkanal uni- oder bidirektional sein. Die Öffnung eines Datenkanals erfolgt durch den Austausch entsprechender H.245-Nachrichten.

Komponenten - Terminal. Ein Terminal ist der Endpunkt einer Kommunikationsverbindung. Es muss mindestens ein Audiokodierungsverfahren, eine Systemkontrolleinheit, eine H.225.0-Schicht und eine Netzwerkschnittstelle verfügen. Video- und Datenanwendungen können optional unterstützt werden.

Komponenten - Gatekeeper. Ein Gatekeeper ermöglicht zusätzliche Dienste für die Verbindungskontrolle zur Unterstützung von H.323-Endpunkten. Die Dienste sind über die RAS-Funktionen in H.225.0 definiert und lassen sich wie folgt einteilen: Adressübersetzung, Zugangskontrolle, Bandbreitenkontrolle. Außerdem kann ein Gatekeeper folgende Dienste optional realisieren: Verbindungskontrolle, Gesprächsautorisierung, Bandbreitenmanagement.

Komponenten - Gateway. Ein H.323-Gateway ist ein Endpunkt im Netzwerk, der Zwei-Wege-Kommunikation zwischen H.323-Terminals in paketbasierten Netzen und anderen ITU-Terminals in leitungsvermittelten Netzen oder zu einem anderen H.323-Gateway ermöglicht. Zu den anderen ITU-Terminals gehören z.B. H.320 (ISDN).

Komponenten - Multipoint Control Unit. Eine Multipoint Control Unit (MCU) bietet Unterstützung für Multipoint-Konferenzen. Durch die entsprechenden Prozeduren können H.323-Endpunkte Verbindungen zu einer MCU aufbauen. Die MCU sorgt dann für die Kopplung der verwendeten Medien (z.B. Mischen der Audioströme aller Konferenzteilnehmer).

H.323-Szenario. In Abbildung 11 ist ein typisches IP-Telefonieszenario, basierend auf der H.323-Protokollfamilie, dargestellt. Das Bild zeigt das private Netz einer Organisation, das zum Internet hin durch eine Firewall geschützt wird. Innerhalb dieses Intranets existiert eine (alternativ auch mehrere) H.323-Zone, die einen Gatekeeper und mehrere optionale Geräte, wie zum Beispiel eine MCU, Gateways und Terminals umfasst.

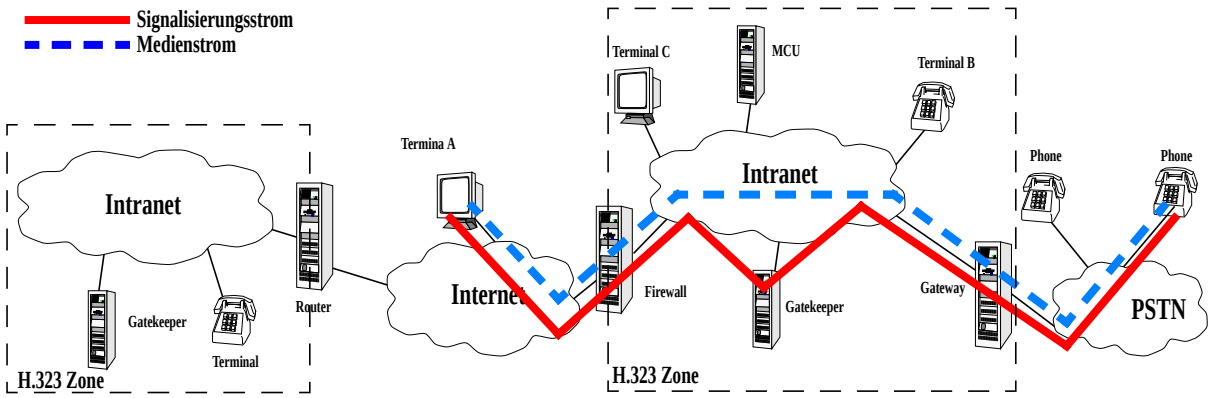


Abbildung 11: H.323-Szenario

Ein solches IP-Telefonie Szenario besitzt eine Signalisierungs- und eine Medientransportebene sowie verschiedene Telefoniekomponenten. Die Signalisierungsebene wird verwendet, um die notwendigen Signalisierungsinformationen zwischen den Komponenten zu transportieren. Nachdem ein Ruf aufgebaut wurde, wird die Medientransportebene dazu benutzt, die Sprachdaten zwischen den Komponenten, z.B. Terminals oder Gateways zu befördern. Oft erfolgt die Konfiguration der Komponenten über Fernzugriff, so dass auch Verwaltungsdaten transportiert werden müssen. Dies könnte als zusätzliche dritte Ebene betrachtet werden, oft wird aber diese Funktion als Ergänzung zur Signalebene angesehen.

3.4.2 Angriffe auf H.323-Systeme

Für die Nutzung eines H.323-Dienstes in einem über einen Experimentalbetrieb hinausgehenden Grad ist die Sicherheit entsprechender Lösungen eine wichtige und zu hinterfragende Anforderung. Für die vorliegende Arbeit ist es insbesondere notwendig zu ermitteln, welche Sicherheitsanforderungen durch eine Firewall unterstützt und in welcher Weise erbracht werden müssen.

Da mittlerweile von einer Reihe von Herstellern entsprechende Lösungen verfügbar sind, wurden entsprechende Überlegungen zur formalen Klassifizierung von Angriffspunkten und Verletzlichkeiten sowie praktische Experimente zu deren Ausnutzung ausgeführt und sind hier im nachfolgenden Abschnitt beschrieben. Eine ausführliche Beschreibung dieser Sachverhalte ist in [42] und [43] gegeben.

Generelle Angriffsmöglichkeiten auf H.323-Systeme. Für eine Sicherheitsbewertung von H.323-Szenarien muss eine Reihe von korrespondierenden Fakten berücksichtigt werden. Zunächst sind alle verwendeten Ebenen abhängig von derselben Infrastruktur, dem IP-Netzwerk. Gefährdungen auf Signalisierungsebene, Medienebene und Verwaltungsebene können daher auf unzuverlässigen Netzwerkteilen, Komponenten oder Betreibern beruhen. Letztlich wird die Infrastruktur auch von anderen Diensten genutzt, und es kann daraus geschlossen werden, dass nicht nur telefoniebezogene Sicherheitsprobleme auftreten können. Generell können die möglichen Gefährdungen und die davon abgeleiteten Angriffspunkte folgendermaßen klassifiziert werden:

- **Signalisierungsebene, Medientransportebene, Verwaltungsebene:** In erster Linie können die H.323-Systeme selbst auf den verschiedenen Ebenen angegriffen werden. Signalisierungsebene, Medientransportebene und Verwaltungsebene können dabei Ziel eines Angriffs sein.
- **Umgebung:** Die Umgebung, die für einen spezifischen H.323-Dienst verwendet wird, kann Ziel eines Angriffs sein. Dazu gehören Betriebssysteme, die die H.323-Komponenten tragen. Diese Betriebssysteme selbst sind natürlich ebenfalls angreifbar.
- **Sicherheitssysteme:** Vorhandene Sicherheitssysteme, die nicht unmittelbar dem H.323-System zugerechnet werden, können angegriffen werden. So könnte z.B. eine von einem H.323-Gespräch durchlaufene Firewall geschwächt werden, weil sie H.323-Kommunikation unterstützt und zeitweise gewisse Kommunikationswege öffnet, die dann nicht für die normale erwünschte Konversationen, sondern für Angriffe verwendet werden können.
- **Menschen:** Da die meisten Systeme, wie auch ein H.323-System, von Menschen verwendet wird, bietet sich auch hier eine Angriffsfläche.

Der folgende Abschnitt gibt eine Auswahl der untersuchten Verletzlichkeitsbeispiele, die entsprechend der Klassifizierung eingeordnet werden können. Bei den gegebenen Beispielen handelt es sich um Angriffe, die sich in die erste Kategorie (Angriffe auf Signalisierungsebene, Medientransportebene, Verwaltungsebene) einordnen lassen. Angriffe, die sich in die restlichen Kategorien einordnen lassen, sind nicht H.323 spezifisch - solche Angriffsmöglichkeiten treten bei allen vernetzten Systemen auf - und werden deshalb in dieser Arbeit nicht untersucht.

Angriffsmöglichkeiten. Im Folgenden werden einige exemplarische Angriffe auf verschiedene Elemente eines H.323-Szenarios (siehe Abbildung 11) dargestellt. Die dargestellten Angriffe beziehen sich auf die für die Tests verwendeten herstellerspezifischen Komponenten. Es hat sich aber gezeigt, dass eine Vielzahl der beschriebenen Angriffe oft auf die verschiedenen Implementierungen der Hersteller anwendbar sind.

- **DoS-Angriff auf ein Terminal über die H.323-Signalisierung:** Die ausgewerteten Ensysteme waren nicht in der Lage, einem Angriff, bei dem unerwartete oder inkorrekte H.323-Signalisierungs-PDUs versandt wurden, zu widerstehen. Dies hatte zur Folge, dass ein System entweder zeitweise nicht verfügbar war, oder dass das System sogar ganz ausfiel, weil das Gerät blockierte, abstürzte oder neu startete.
- **DoS-Angriff auf ein Terminal über die administrative Schnittstelle:** Das untersuchte IP-Telefon benutzt einen integrierten WWW-Server, mit dem das Gerät verwaltet und einige seiner Einstellungen abgefragt werden können. Dieser WWW-Server und seine Implementierungsmängel (obwohl nicht grundlegend verantwortlich für die IP-Telefoniefunktionen des Gerätes) machen es für Angriffe anfällig. Wenn an den integrierten WWW-Server eine ausreichend lange URL versandt wird, kann das Gerät (abhängig von der Länge des URL-Strings) entweder außer Betrieb gesetzt werden oder es wird neu gestartet.
- **Übernahme eines Terminals über die administrative Schnittstelle:** Die Schnittstelle für die Fernverwaltung über HTTP, die das IP-Telefon verwendet, ist für Angriffe anfällig. Das Administratorpasswort wird im Klartext versandt, wodurch dieses abhörbar ist. Darüber hinaus kann das Administratorpasswort auch anhand einer Reihe von automatisierten Brute Force Versuchen angegriffen werden. Eine Begrenzung der Rateversuche erfolgt nicht, wegen seiner begrenzten Länge und eingeschränktem Alphabet - da das Passwort normalerweise über die Telefontastatur eingegeben werden muss - ist dieses auch leicht zu erraten. Damit erhält der Angreifer wiederum vollen Zugriff auf alle Konfigurationsmöglichkeiten des Geräts.
- **Angriff auf die Vertraulichkeit der H.323-Medienströme:** H.323-Anwendungen benutzen mit dem UDP-Protokoll versandte RTP-Pakete, um Audioströme zu befördern. Ein Angreifer, der Zugang zu dem verwendeten Kommunikationsweg hat, muss die Datenströme, die die Audioverbindung(en) repräsentieren, zunächst identifizieren. Danach ist ein Abhören bzw. Aufzeichnen von Gesprächen ohne Probleme möglich.
- **DoS-Angriffe auf Gatekeeper über die H.323-Signalisierung:** Es konnten alle betrachteten Gatekeeper daran gehindert werden, ihre normalen Aufgaben auszuführen, indem ihnen eine große Anzahl von entweder regulären (zyklischen Endgeräteanmeldungen bzw. -abmeldungen) oder irregulären H.323-PDUs zugesendet wurde. Dadurch steht der H.323-Dienst entweder nur für eine gewisse Zeit oder überhaupt nicht mehr zur Verfügung (falls der Gatekeeper zum Absturz gezwungen wurde).

- **Angriff auf die Gatekeeper Anmeldung über die H.323-Signalisierung:** Während der Anmeldung an einem Gatekeeper registriert ein H.323-Terminal seine IP-Adresse, seine E.164-Nummer und eine beliebige Anzahl zusätzlicher symbolischer Namen (sogenannte Aliase). Eine Abmeldung eines angemeldeten Terminals ist durch Fälschen einer RAS-Meldung von beliebiger anderer Stelle im Netzwerk möglich. Dadurch kann das Terminal keine Verbindungen mehr entgegen nehmen oder selbst absetzen. Des Weiteren kann der Angreifer sich nun unter der E.164-Nummer des angegriffenen Terminals registrieren und unter dieser Identität Gespräche führen und empfangen.

Maßnahmen. Die zu ergreifenden Maßnahmen beziehen sich auf die verschiedensten Punkte innerhalb des H.323-Szenarios. Im Folgenden wird betrachtet, welche der oben beschriebenen Angriffe sich durch die Maßnahme “Firewall” am Übergang zwischen internem und externem Netz neutralisieren lassen. Prinzipiell kann eine Firewall nur dann von Nutzen sein, wenn die betrachtete Kommunikation über sie abgewickelt wird. Angriffe, bei denen nur interne Komponenten beteiligt sind, können durch die Maßnahme “Firewall” nicht unterbunden werden.

- **DoS Angriff auf ein Terminal/Gatekeeper über die H.323-Signalisierung, Angriff auf die Gatekeeper Anmeldung über die H.323-Signalisierung:**
  Alle H.323-Signalisierungs-PDUs können geprüft werden, bevor diese von der Firewall weitergeleitet werden. Dabei kann geprüft werden, ob die gesendeten PDUs
  - eine korrekte Struktur besitzen, um zu verhindern das irregulären H.323-PDUs weitergeleitet werden.
  - in den Kontext der aktuell stattfindenden Kommunikation passen, um unerwartete PDU-Abfolgen zu verhindern.
  - überhaupt weitergeleitet werden sollen. Durch Verwerfen der PDU kann verhindert werden, dass bestimmte Funktionen über die Netzgrenze hinweg nutzbar sind.

  Werden PDUs für einen Angriff verwendet, die durch die oben beschriebenen Maßnahmen nicht als PDUs eines Angriffs identifiziert werden können, müssen weitere Funktionen innerhalb einer Firewall realisiert werden. Über die Integration eines Intrusion Detection Systems (IDS) in eine Firewall können teilweise Angriffe über reguläre Protokollabläufe erkannt und verhindert werden ([44], [45]). Das Prüfen der Signalisierungssemantik erfordert es, dass die Firewall über eine korrekte H.323-Implementierung verfügt und in der Lage ist, die Signalisierung auf Applikationsebene zu verarbeiten.
- **DoS Angriff auf ein Terminal über die administrative Schnittstelle, Übernahme eines Terminals über die administrative Schnittstelle:**
  Eine Firewall ist in der Lage bestimmte Kommunikationswege zu blockieren. Da auf die administrative Schnittstelle eines Terminals in der Regel nicht von außerhalb zuge-

griffen werden muss, können alle Zugriffe auf diese Schnittstelle von außerhalb präventiv unterbunden werden.

Zusammenfassung. Die hier dargestellte Untersuchung zeigt, dass eine Sicherheitsüberprüfung der Signalisierung auf Applikationsebene durch eine Firewall vorgenommen werden muss[1]. Dadurch können viele mögliche Angriffe - vorausgesetzt die Angriffe verwenden einen Kommunikationsweg, der über eine Firewall läuft - verhindert werden. Es zeigt sich ebenfalls, dass eine Sicherheitsüberprüfung der Medienströme auf Applikationsebene nicht notwendig ist, da dadurch zumindest keine der hier dargestellten Angriffe verhindert werden können. Die Untersuchung zeigt, dass die hier gewonnenen Erkenntnisse, die in Abschnitt 3.3.1 getroffenen Annahmen, bestätigen. Die hier gewonnenen Erkenntnisse werden auch durch die in [46] aufgestellte Sammlung bekannter Bedrohungen in H.323-Szenarien bestätigt.

3.4.3 H.323-Applikationen und Firewalls

Wie in Abschnitt 3.2.2 beschrieben, sind bestimmte Charakteristika von Multimedia-Applikationen dafür verantwortlich, dass deren Verwendung in einer Umgebung, in der auch Firewalls eingesetzt werden, problematisch ist. Diese Charakteristika bzw. die daraus resultierenden Probleme, sind auch in H.323-Szenarien zu finden. Im Folgenden werden diese Probleme genauer erläutert.

Um nachfolgend die durch die einzelnen Charakteristika hervorgerufenen Probleme verständlich erläutern zu können, ist es nötig, die H.323-Kommunikation genauer zu beschreiben. Dies kann innerhalb dieser Arbeit nicht in vollem Umfang geschehen (Siehe [12] für eine genaue Beschreibung aller möglichen Fälle). Es wird dazu die im Folgenden erklärte vereinfachte Darstellung, die zur Erläuterung der folgenden Abschnitte dient, verwendet.

Wenn nur zwei Terminals an der Kommunikation beteiligt sind (direkter Ruf zwischen Terminal A und Terminal B), wird der in Abbildung 12 dargestellte Kommunikationsablauf verwendet. Die Pfeile stellen dabei die Richtung der Kommunikationsinitiierung dar. Bei TCP bedeutet dies, dass in Richtung des Pfeiles die Verbindung aufgebaut wird. Daten aber werden in beide Richtungen gesendet. Bei UDP wird in Richtung des Pfeiles das erste Paket des Stroms transportiert. Die Abfolge der Antworten in Gegenrichtung (unter Verwendung derselben Ports) ist aus Gründen der Vereinfachung nicht vollständig aufgeführt. Für die Definition der Security Policy innerhalb einer Firewall ist es notwendig zu wissen, wer die Verbindung initiiert und welche Ports dabei verwendet werden. Daher ist eine solche Darstellung hier besonders geeignet.

[1] In Abschnitt 3.4.3 wird gezeigt, dass die Signalisierung ebenfalls auf Applikationsebene verarbeitet werden muss, um bestimmte, die Eigenschaften der Applikation betreffende, Informationen (Portnummern, IP-Adressen) zu erhalten.

- **Q.931 Call Signalling (TCP):** Eine TCP-Verbindung wird zwischen Terminal A und Terminal B aufgebaut. Diese wird zum Transport der *Call Setup* Nachrichten, wie sie in H.225.0 definiert sind, verwendet. Diese Nachrichten werden benötigt, um den Rufaufbau durchzuführen. Dabei werden unter anderem die Parameter (Port und IP-Adresse) für die folgende Call Control Verbindung an Terminal A übermittelt.
- **H.245 Call Control (TCP):** Terminal A kontaktiert Terminal B via TCP unter Verwendung der zuvor übermittelten Parameter (Port und IP-Adresse). Die H.245-Verbindung wird verwendet, um *Call Control* Nachrichten zwischen den Terminals auszutauschen. Diese Nachrichten werden unter anderem dazu verwendet, die Parameter der nachfolgenden Medienströme auszuhandeln (*OpenLogicalChannel*). Dabei werden neben den entsprechenden Medienkodierungsverfahren auch die zu verwendenden IP-Adressen und UDP-Ports vereinbart.
- **RTP/RTCP Media und Mediacontrol (UDP):** Zwischen den beiden Terminals werden mehrere Medienströme verwendet. Es sind mindestens 4 UDP-Ströme notwendig, um die Audiodaten zu transportieren (Ein RTP- und der korrespondierende RTCP-Strom in jede Richtung). Zusätzliche Ströme werden verwendet, wenn zum Beispiel eine optionale Videoübertragung stattfindet.

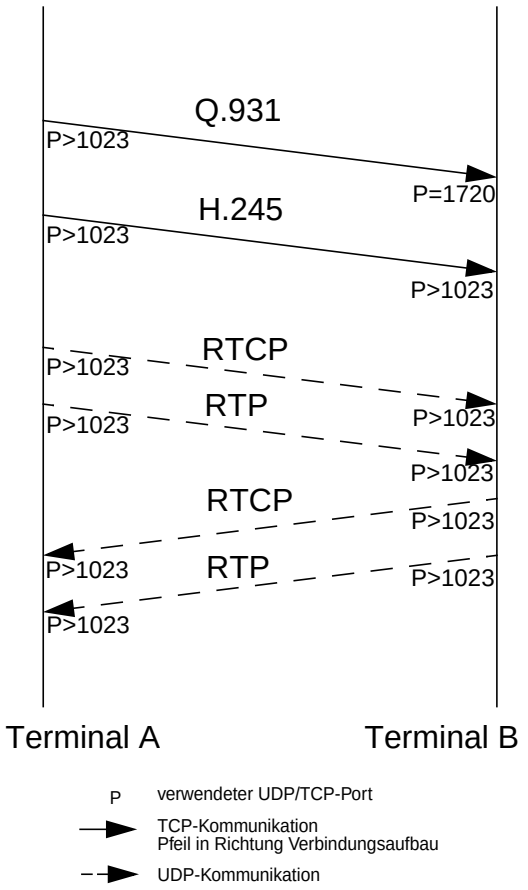


Abbildung 12: H.323 direkter Ruf

Zum Beenden der Verbindung werden über den Call Conrol Kanal entsprechende Meldungen ausgetauscht, danach werden die Medienströme abgebaut und die Kontrollkanäle werden geschlossen.

Protokollcharakteristika - Komplexität der Protokolle. Wie in Abschnitt 3.4.1 beschrieben, stellt das H.323-Protokoll eine Protokollfamilie dar. Innerhalb des H.323-Standards sind für einzelne Teilbereiche weitere Protokolle definiert bzw. angegeben (z.B. H.225, H.245, RTP/RTCP, usw.). Dementsprechend verwendet eine H.323-Komponente einen Teil - oder sogar alle - der innerhalb des H.323-Standards beschriebenen Protokolle, um ihre vorgesehene Aufgabe innerhalb eines H.323-Szenarios zu erfüllen. Daraus ergibt sich, dass eine H.323-Komponente zur Erfüllung ihrer Aufgabe verschiedene Protokollautomaten bzw. Parser besitzt. Die einzelnen, innerhalb einer Komponente verwendeten Protokollautomaten und Parser sind zusätzlich untereinander verbunden. Die Zustände der einzelnen Protokollautomaten hängen jeweils vom Zustand

der anderen Protokollautomaten ab. Beispielsweise ist der Zustand des Q.931-Protokollautomaten vom Zustand des H.245-Protokollautomaten abhängig und umgekehrt. Eine Firewall, die innerhalb eines H.323-Szenarios eingesetzt wird, muss in der Lage sein, solche komplexen und mehrdimensionalen Zustandsautomaten zu implementieren und zu verwalten. Dies ist notwendig, um alle zu einer Sitzung gehörenden Ströme zu identifizieren. Die hohe Komplexität des H.323-Protokolls macht innerhalb der Firewall ein ebenso komplexes Element zur Verarbeitung des H.323-Protokolls notwendig.

Bestehende Firewalls sind in der Regel nicht dafür ausgelegt solch komplexe Zustandsautomaten zu implementieren. Es kann vorkommen, dass die notwendigen Zustandsautomaten nicht mehr in ein bestehendes System integriert werden können, da die verfügbaren Ressourcen (z.B. Speicherplatz, Prozessorleistung) für die Abbildung dieser komplexen Zustandsautomaten nicht ausreichen.

Ein weiteres Problem stellt die Verwendung einer ASN.1 Kodierung für die Nachrichten verschiedener Teilprotokolle (H.225, H.245) dar. Durch die Verwendung von ASN.1 werden die zu übermittelnden Informationen sehr dicht in die Nachrichten gepackt, wodurch die vorhandenen Übertragungskapazitäten sehr effizient genutzt werden können. Allerdings müssen die Nachrichten kodiert und dekodiert werden. Dazu muss eine H.323-Element innerhalb der Firewall in der Lage sein, die ASN.1 Kodierung bzw. Dekodierung vornehmen zu können. Die dazu notwendigen Funktionen und Mechanismen erhöhen zusätzlich die Komplexität des Elements zur H.323-Verarbeitung innerhalb der Firewall.

Protokollcharakteristika - Mehrere Ströme in einer Sitzung. Ein wesentliches Merkmal von Multimedia-Protokollen ist die Verwendung mehrerer Ströme, die zusammen eine logische Sitzung bilden. Dieses Merkmal ist innerhalb von H.323 besonders stark ausgeprägt. Im Unterschied zu anderen Multimedia-Protokollen (siehe Abschnitt 2.3) besteht eine H.323-Sitzung aus mehreren Signalisierungs- und Medienströmen. Für eine Firewall stellt sich nun das Problem, die verschiedenen Ströme, die zusammen eine Sitzung bilden, identifizieren zu können. Diese Identifizierung ist aus zwei Gründen notwendig. Erstens muss eine Firewall in der Lage sein zu identifizieren, welche Kommunikationspartner innerhalb einer Sitzung Daten austauschen. Diese Information stellt die Grundlage der sicherheitsrelevanten Entscheidung (z.B. ob die entsprechenden Daten weitergeleitet werden) innerhalb einer Firewall dar. Zweitens muss die Firewall in der Lage sein, den (zeitlichen) Zusammenhang der auftretenden Ströme innerhalb einer Sitzung zu prüfen, um zu verhindern, dass ein Strom von einem Angreifer in einem ungültigen Kontext verwendet wird.

Die Erkennung dieser Zusammenhänge stellt an eine Firewall besondere Anforderungen. Es ist erforderlich, dass die Firewall fähig ist die Teile des zwischen den Kommunikationspartnern ablaufenden H.323-Protokolls zu interpretieren, die Aufschluss über die aktuell verwendeten Ströme einer Sitzung geben. Dies betrifft innerhalb des H.323-Protokolls das H.225/Q.931-Protokoll und das H.245-Protokoll. Mit Hilfe des Q.931-Protokolls werden Informationen zwischen den Kommunikationspartnern ausgetauscht, die dazu verwendet werden den H.245-Signalisierungsstrom zu etablieren. Dies geschieht über die Q.931-Nachricht *Setup*. Es besteht die Möglich-

keit, auf den zweiten Signalisierungsstrom zu verzichten und die H.245-Nachrichten über den gleichen Kanal zu transportieren, über den auch die Q.931-Nachrichten ausgetauscht werden (bezeichnet als *H.245Tunneling*). Damit eine Firewall den Unterschied zwischen diesen beiden Operationsmodi erkennen kann, muss innerhalb der Q.931-Nachrichten das Feld *H.245Tunneling* überprüft werden. Innerhalb der dann folgenden H.245-Nachrichten werden zwischen den Kommunikationspartnern die notwendigen Medienströme vereinbart. Je nach Beschaffenheit der Kommunikationspartner kann es sich um Video-, Audio- und Datenströme handeln. Die entsprechenden Ströme werden durch den Austausch von *OpenLogicalChannel, OpenLogicalChannelAck* bzw. *OpenLogicalChannelReject* H.245-Nachrichten festgelegt. Während einer H.323-Sitzung können verschiedene Ströme neu zu einer Sitzung hinzukommen, bzw. bestehende abgebaut werden. Dazu können ebenfalls die oben beschriebenen H.245-Nachrichten verwendet werden. Bei der Beendigung der H.323-Sitzung werden zuerst die Medienkanäle geschlossen. Dies geschieht durch den Austausch von *CloseLogicalChannel* H.245-Nachrichten. Die jeweiligen Nachrichten werden durch entsprechende *CloseLogicalChannelAck* bestätigt. Danach wird der H.245-Kanal, falls nicht *H.245Tunneling* verwendet wird, geschlossen. Als Letztes wird dann der Q.931-Kanal terminiert, dazu wird die *ReleaseComplete* Q.931-Nachricht ausgetauscht und anschließend der Kanal geschlossen. Es ist ebenfalls möglich, dass Medienströme schon innerhalb des initialen Austausches von Q.931-Nachrichten vereinbart werden (bezeichnet als *Faststart*). Dieser Operationsmodus muss, wenn verwendet, von einer Firewall ebenfalls berücksichtigt werden.

Die genaue Abfolge der einzelnen Nachrichten, sowie die möglichen Varianten (*H245Tunneling, Faststart*), die für einen H.323-Sitzungsaufbau verwendet werden, sind in [12] genau dargestellt. Eine H.323-Firewall muss, um die verschiedenen Ströme einer Sitzung zuordnen zu können, mindestens diese Nachrichten interpretieren können. Darüber hinaus muss die Firewall entsprechende Zustände über den Kontext der Nachrichten abspeichern. Dies bedeutet letztendlich, dass die Firewall für das Q.931- sowie für das H.245-Protokoll einen Protokollautomaten implementieren muss, der zumindest auf den oben beschriebenen Nachrichten basieren muss. Eine H.323-Firewall benötigt dementsprechend zumindest einen reduzierten H.323-Protokoll-Stack. Tabelle 1 fasst zusammen, welche Nachrichten im Wesentlichen zur Identifizierung der einzelnen Ströme von einer Firewall interpretiert werden müssen.

Tabelle 1: H.323-Nachrichten zur Beschreibung der verwendeten Ströme

Protokoll	Nachrichten	Beschreibung
Q.931	*Setup*	Vereinbarung des Call Control Kanals (H.245)
Q.931	*ReleaseComplete*	Aufheben des Call Signalling Kanals, sowie der davon abhängigen Kanäle Call Control, Media und Media Control (Q.931, H.245, RTP/RTCP))
Q.931	*(H.245Tunneling)* [a]	Tunneln der Call Control-Nachrichten über den Call Signalling Kanal

Tabelle 1: H.323-Nachrichten zur Beschreibung der verwendeten Ströme

Protokoll	Nachrichten	Beschreibung
Q.931	*(Faststart)*[b]	Vereinbarung der Media und Media Control Ströme innerhalb von Call Signalling-Nachrichten
H.245	*OpenLogicalChannel*, *OpenLogicalChannelAck*, *OpenLogicalChannelReject*	Vereinbarung der Media und Media Control Ströme (RTP/RTCP)
H.245	*CloseLogicalChannel*, *CloseLogicalChannelAck*	Aufheben der Media und Media Control Ströme (RTP/RTCP)

a. Diese Option wird in verschiedenen Nachrichten übermittelt
b. Diese Parameter werden innerhalb verschiedener Q.931 Nachrichten übermittelt

Es existieren weitere Nachrichten die ebenfalls zur Steuerung der einzelnen Ströme verwendet werden können (z.B. *requestChannelClose*). Alle Nachrichten und Operationsmodi sind genau in [12] beschrieben.

Die hier beschriebenen Nachrichten enthalten neben der Information, dass ein bestimmter Kanal verwendet werden soll, ebenfalls Informationen über die Charakteristika der entsprechenden Kanäle. Diese dynamisch festgelegten Charakteristika werden im folgenden Abschnitt erläutert.

Protokollcharakteristika - Dynamisches Verhalten. Multimedia-Protokolle besitzen viele dynamische Anteile; dies gilt insbesondere für das H.323-Protokoll. Wie bereits erwähnt, werden die Charakteristika der verwendeten Ströme größtenteils dynamisch festgelegt. Die dynamischen Festlegungen schließen das verwendete Protokoll, die verwendeten Quell- und Ziel-Ports sowie die Quell- und Zieladressen ein. Darüber hinaus wird für die Medienströme dynamisch festgelegt, welche Bandbreite diese verwenden. Weitere Dynamik entsteht durch die Tatsache, dass die Anzahl der Ströme sich während der Dauer einer Sitzung ändern kann. Die wesentlichen Nachrichten zur Übermittlung der dynamischen Parameter sind ebenfalls in Tabelle 1 zusammengefasst.

Diese Dynamik erschwert es der Firewall zusätzlich, die Zuordnung der Ströme zu ihrer entsprechenden Sitzung vorzunehmen.

Applikationscharakteristika - Szenarienvielfalt und Szenarienkomplexität. Die verwendeten Kommunikationsmechanismen innerhalb eines H.323-Szenarios sind von den in die Kommunikation involvierten Geräten abhängig und ändern sich je nach Anwendungsfall. Wenn nur zwei Terminals an der Kommunikation beteiligt sind (direkter Ruf zwischen Terminal A und Terminal B), wird der in Abbildung 12 dargestellte Kommunikationsablauf verwendet.

Wenn innerhalb dieses Szenarios ein Gatekeeper verwendet wird (Rufvermittlung durch den Gatekeeper), ändern sich die Kommunikationsbeziehungen und Kommunikationsabläufe. In dem hier dargestellten Fall (Abbildung 13) laufen die Signalisierungsnachrichten über den Gatekeeper

(*gatekeeper routed call*). Der H.323-Standard sieht auch eine Möglichkeit vor, den Gatekeeper ohne direkte Einbeziehung in die Signalisierung zu verwenden (*direct call model*).

Der *gatekeeper routed call* wird jedoch in den meisten Gatekeeper-Szenarien verwendet.

- **RAS Registration, Admission, Status (UDP):** Nach dem Start der Terminals registrieren sich diese am Gatekeeper (*RegistrationRequest RRQ* und *RegistrationConfirm RCF*). Dazu wird das RAS-Protokoll verwendet, welches UDP als Transportprotokol benutzt. Den Terminals muss dazu die Adresse des Gatekeepers bekannt sein; in diesem Beispiel wird eine statische Konfiguration angenommen.
- **RAS Registration, Admission, Status (UDP):** Bevor die Kommunikation stattfinden kann, muss das rufende Terminal beim Gatekeeper unter Verwendung des RAS-Protokolls mit einer Spezifikation der geplanten Gesprächscharakteristika eine entsprechende Erlaubnis erbitten (*AdmissionRequest ARC*). Wenn die Erlaubnis erteilt wird (*AdmissionConfirm ACF*), kann der eigentliche Ruf ausgelöst werden.
- **Q.931 Call Signalling (TCP):** Terminal A kontaktiert den Gatekeeper via TCP. Anhand der ersten Call Signalling Nachricht kann der Gatekeeper das Zielterminal bestimmen und kontaktiert dieses. Das Zielterminal führt daraufhin ebenfalls eine Erlaubnisanfrage durch, ob es den Ruf annehmen darf. Die Call Signalling Nachrichten werden in diesem Fall über den Gatekeeper geroutet. Die Parameter, der Port und die IP-Adresse für die folgende Call Control Verbindung werden an Terminal A übermittelt (innerhalb der Call Signalling Nachricht *Setup*).

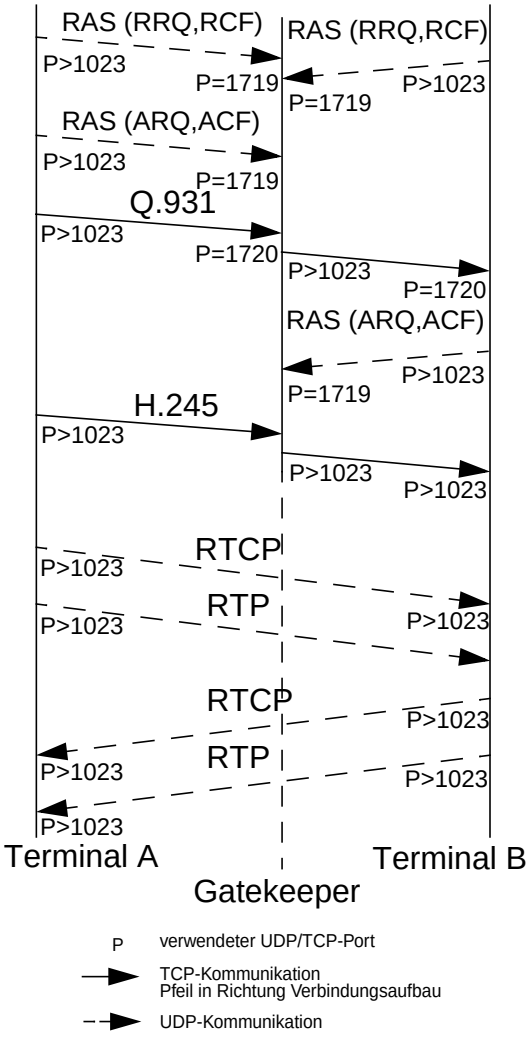


Abbildung 13: H.323 Gatekeeper vermittelter Ruf

- **H.245 Call Control (TCP):** Terminal A kontaktiert den Gatekeeper via TCP unter Verwendung der zuvor ausgehandelten Parameter. Der Gatekeeper kontaktiert seinerseits das Terminal B. Die Call Control Nachrichten werden in diesem Fall ebenfalls über den Gatekeeper geroutet. Die *Call Control* Nachrichten werden dazu verwendet, die Parameter der folgenden Medienströme auszuhandeln.

- **RTP/RTCP Media und Media Control (UDP):** Wie im ersten Beispiel werden nun mehrere Medienströme zwischen beiden Terminals verwendet. Diese werden direkt gesendet, ohne den Gatekeeper in die Kommunikation einzubeziehen.

Die Kommunikationsabläufe ändern sich ebenfalls, wenn Protokollvarianten wie *H.245Tunneling* oder *Faststart* verwendet werden, bzw. weitere H.323-Komponenten in die Kommunikation mit einbezogen sind. Es zeigt sich, dass sich das Kommunikationsverhalten signifikant ändern kann, wenn sich das Anwendungsszenario ändert. Eine sich innerhalb der Kommunikationswege befindende Firewall muss dieser dynamischen Vielfalt gerecht werden.

Applikationscharakteristika - Herstellerspezifische Implementierungen. Nicht nur die Nutzung verschiedener H.323-Komponenten sondern auch deren unterschiedlichen herstellerspezifischen Implementierungen haben einen Einfluss auf die Kommunikationsabläufe. Die Experimente zeigen, dass unterschiedliche Hersteller auch unterschiedliche (und teilweise sogar nicht interoperable) Implementierungen verwenden, obwohl alle angeben, H.323-kompatibel zu sein.

Wie bereits beschrieben, werden die verwendeten Medienströme über entsprechende H.245-Nachrichten zwischen den Kommunikationspartnern ausgehandelt. Ein Terminal, das beispielsweise einen Videostrom an das andere Terminal übermitteln möchte, sendet dazu eine entsprechende *OpenLogicalChannel* Nachricht an das andere Terminal. Dieses bestätigt die *OpenLogicalChannel* Nachricht mit einer entsprechenden *OpenLogicalChannelAck* Nachricht. Danach sind die Stromparameter festgelegt und der Videostrom kann zwischen den Terminals ausgetauscht werden. Werden mehrere Ströme verwendet, wiederholt sich dieser Vorgang. Je nach Art der verwendeten Terminals können diese Nachrichten aber auch in anderer Reihenfolge ausgetauscht werden. Beispielsweise können die Terminals zuerst alle nötigen *OpenLogicalChannel* Nachrichten austauschen und danach alle *OpenLogicalChannelAck* Nachrichten. In anderen Fällen wird nach jeder *OpenLogicalChannel* Nachricht zunächst ein *OpenLogicalChannelAck* gesendet. Für eine dazwischenliegende Firewall ergibt sich nun das Problem, dass sie mit jeder Variation umgehen können muss.

Ein weiteres Beispiel ist in der Q.931 *Setup* Nachricht zu finden. Wie bereits erwähnt enthält die *Setup*-Nachricht die Adresse des gerufenen Terminals. Der H.323-Standard lässt mehrere Möglichkeiten zu diese Zieladresse innerhalb der Nachricht unterzubringen (im Feld *CalledPartyNumber* oder innerhalb der *UserUserIE* im Feld *DestinationAddress*). Manche Terminals verwenden das *CalledPartyNumber*-Feld, andere das *DestinationAddress*-Feld und wieder andere Terminals füllen die Information in beide Felder. Eine dazwischenliegende Firewall muss in der Lage sein alle Varianten zu unterstützen um ein flexibles Call Routing zu ermöglichen.

Applikationscharakteristika - Verwendung zusätzlicher Protokolle. Ist zum Beispiel Terminal A kein reines H.323-Terminal sondern ein Microsoft Netmeeting Terminal, so kann eine ILS/LDAP-Erweiterung verwendet werden. Bevor die Kommunikation beginnt, versucht das rufende Terminal einen Kontakt zu einem ILS-Server aufzubauen, um eine Namensauflösung durchzuführen. Auf diesem Weg können symbolische Namen in Client IP-Adressen aufgelöst

werden (Telefonbuchfunktion). Nachdem das Terminal die Zieladresse bestimmt hat, startet die normale H.323-Kommunikation. Die Kommunikationsabläufe entsprechen dabei dem direkten Ruf. In diesem Fall werden teilweise die Funktionen des Gatekeepers durch den ILS-Server erbracht. Die dabei verwendeten Kommunikationsmechanismen sind außerhalb des H.323-Standards festgelegt, sind aber logisch der H.323-Kommunikation zuzurechnen.

Bei der Verwendung dieser speziellen H.323-Applikation muss eine im Kommunikationsweg befindliche Firewall in der Lage sein, die für die Kommunikation mit dem ILS Server verwendeten Kanäle und die H.323 spezifischen Kanäle einer logischen Sitzung zuzuordnen.

Leistungscharakteristika. Um eine H.323-Applikation sinnvoll zu betreiben, sind verschiedene Dienstgüteanforderungen zu erfüllen. Hinsichtlich der Unterstützung der H.323-Applikationen durch Firewalls ist die durch die Firewall zusätzlich eingebrachte Verzögerung der Signalisierungs- und Mediendaten sowie der erreichbare Datendurchsatz wesentlich. Diese Aspekte werden nachfolgend behandelt.

Die Signalisierung wird dazu verwendet, die Medienströme auszuhandeln und zu initiieren. An die Geschwindigkeit, mit der diese Aushandlung geschieht, sind gewisse Mindestanforderungen zu stellen. Folgendes Beispiel soll dies verdeutlichen: Ein Terminal A ruft (direkter Ruf) ein Terminal B an. Die Q.931 Call Signalling-Verbindung wird zwischen beiden Terminals aufgebaut und die zur Initiierung notwendigen Q.931-Nachrichten werden ausgetauscht. Zu diesem Zeitpunkt beginnt das Terminal B zu klingeln, um anzuzeigen das ein Verbindungswunsch besteht. Nimmt nun eine Person an Terminal B den Ruf entgegen, z.B. durch Abheben des Hörers, so wird die H.245 Call Control Verbindung aufgebaut. Über die H.245 Call Control Verbindung werden nun die Medienströme ausgehandelt und initiiert. Zur Aushandlung und Initiierung der Medienströme bleibt in diesem Beispiel gerade soviel Zeit, wie die Person an Zeit benötigt, um den abgehobenen Telefonhörer zum Ohr zu führen. Ist dies geschehen, wird die Person sich in der Regel als erstes melden, z.B. mit dem eigenen Namen. Sind zu diesem Zeitpunkt die Medienströme noch nicht geschaltet, werden diese ersten Worte verlorengehen, was zu erheblichen Irritationen der Gesprächsteilnehmer führen kann. Um diesem Effekt vorzubeugen, wurde innerhalb von H.323 die *Faststart*-Variante festgelegt. Diese ermöglicht eine Festlegung der Medienströme schon innerhalb des Q.931-*Setup*. Allerdings wird diese Variante zur Zeit nicht von allen Terminaltypen unterstützt. Eine Firewall sollte dementsprechend eine möglichst geringe zusätzliche Verzögerung der Signalisierungsnachrichten verursachen, um den Effekt nicht zu verstärken.

Für über die Firewall transportierte Mediendaten müssen Grenzwerte für die Verzögerung und den Jitter eingehalten werden. Damit Audio- und Videodaten von einem Empfänger verarbeitet werden können, muss hinsichtlich der Ende-Zu-Ende Verzögerung der transportierten Daten ein Grenzwert eingehalten werden. Die Ende-Zu-Ende Verzögerung der Daten wird ohne die Verwendung einer Firewall im Wesentlichen durch das dazwischenliegende Transportnetz verursacht. Eine in den Kommunikationspfad eingebrachte Firewall sollte die durch das Transportnetz verursachte Verzögerung möglichst nicht erhöhen. Da sich die Ende-Zu-Ende Verzögerung der Daten aber zwangsläufig durch die Firewall erhöht,

- sollte dies vorhersagbar geschehen.
  Der Betrag, um den die Ende-zu-Ende Verzögerung der einzelnen Pakete erhöht wird, sollte für jedes Paket möglichst gleich sein. Dadurch wird ein planbares bzw. voraussagbares Verhalten der Firewall gewährleistet. Außerdem wird dadurch verhindert, dass der Jitter durch die Firewall zusätzlich erhöht wird.
- sollte die Erhöhung möglichst gering ausfallen.

Zusätzlich muss eine Firewall den gewünschten Datendurchsatz ermöglichen, damit auch Medienströme mit einer hohen Bandbreite, bzw. mehrere gleichzeitige Sitzungen von einer Firewall gehandhabt werden können.

Probleme bei Network Address Translation (NAT). Weitere Probleme innerhalb eines H.323-Szenarios treten dann auf, wenn eine Adressumsetzung (NAT) von der Firewall durchgeführt wird. Die zuvor beschriebenen H.323-Charakteristika bzw. die dadurch entstehenden Probleme verstärken sich dadurch nochmals. Diese zusätzlichen Schwierigkeiten werden im Folgenden genauer dargestellt.

- **Protokollcharakteristika - Komplexität der Protokolle:** Innerhalb der Applikationsschicht (Schicht 5) der H.323-Protokolle werden IP-Adressen verwendet. Damit NAT für das H.323-Protokoll verwendet werden kann, muss nicht nur die Adressierung der Ströme auf Schicht 3 (IP-Adressen) und Schicht 4 (Port-Nummern) verändert werden, sondern auch alle innerhalb der Schicht 5 übermittelten IP-Adressen sowie Port-Nummern müssen verändert werden. Durch diese zusätzliche Aufgabe erhöht sich die Komplexität der Firewall nochmals.
- **Protokollcharakteristika - Mehrere Ströme in einer Sitzung:** Bei der Ermittlung, welche Ströme einer Sitzung zugeordnet sind, muss die Firewall zusätzlich die aktuell gültige Adressumsetzung beachten.
- **Protokollcharakteristika - Dynamisches Verhalten:** Die verwendeten Adressen der meisten Ströme werden dynamisch ausgehandelt. In dieser Aushandlung muss nun die Adressumsetzung beachtet werden, alle ausgehandelten IP-Adressen und Port-Nummern müssen durch die Firewall modifiziert werden, um die Adressumsetzung entsprechend abbilden zu können.
- **Applikationscharakteristika - Verwendung zusätzlicher Protokolle:** Für die zusätzlich verwendeten Protokolle muss ebenfalls eine Adressumsetzung von der Firewall durchgeführt werden.
- **Leistungscharakteristika - Medien:** In einer Umgebung, in der keine Adressumsetzung nötig ist, kann die H.323-Firewall sich darauf beschränken für die über die Signalisierung vereinbarten Medienströme die entsprechenden Kommunikationswege zu öffnen. Nachdem die entsprechenden Kommunikationswege freigeschaltet sind, kann die Firewall sich darauf beschränken die Medien-Pakete möglichst schnell weiterzuleiten. Wird eine Adressumsetzung notwendig, so muss die Firewall in jedem weiterzuleitenden Paket die Adresse umschreiben. Je nach verwendeter Technik kann

dies zu einem erheblichen Mehraufwand bei der Weiterleitung dieser Pakete führen. Dadurch erhöht sich die Verzögerung sowie der Jitter der entsprechenden Pakete.

3.4.4 Zusammenfassung

Es wurde gezeigt, dass alle allgemein in Abschnitt 3.2.2 aufgestellten und für Firewalls problematischen Charakteristika von Multimedia-Applikationen auch innerhalb von H.323-Applikationen auftreten. Damit besitzen auch die aus diesen Charakteristika abgeleiteten Designprinzipien Gültigkeit für Firewalls, die in einem H.323-Szenario eingesetzt werden.

3.5 SIP-Applikationen

Das Session Initiation Protocol (SIP) wurde anfangs als Mechanismus vorgeschlagen, um Teilnehmer für eine Mbone-Sitzung einzuladen zu können. Das Protokoll wurde davon ausgehend weiterentwickelt und wird zur Zeit verwendet, um für verschiedenste Arten von Sitzungen Teilnehmer einzuladen (z.B. Mbone, Telefonie, Chat).

Das hier beschriebene SIP-Beispiel und das daran anschliessende RTSP-Beispiel wird kürzer als das vorhergehende Beispiel der H.323-Applikationen dargestellt, da erstens viele bereits beschriebene Elemente wiederkehren und zweitens eine ausführliche Beschreibung dem Umfang dieser Arbeit nicht gerecht werden würde. Referenzen auf Literatur mit ausführlichen Beschreibungen wird an den jeweiligen Stellen angegeben.

3.5.1 Grundlagen

Das Session Initiation Protocol wurde im März 1999 als Proposed Standard RFC2543 durch die IETF verabschiedet. Der SIP-Standard beschreibt im Wesentlichen die Basisifunktionalität des Protokolls. Darüber hinaus existieren weitere RFCs und Internet Drafts, die Erweiterungen und Ergänzungen beschreiben. SIP wird verwendet, um eine Multimedia-Sitzung aufzubauen, zu verändern und zu beenden. Da SIP unabhängig vom Typ der Multimedia-Sitzung verwendet wird, kann dieses Protokoll für Telefonie-, Videokonferenz-, Spiele- und andere Sitzungsarten verwendet werden. SIP wird verwendet, um Sitzungsbeschreibungen unter den möglichen Teilnehmern zu verteilen. Zur Beschreibung einer Audio- oder Videositzung wird in der Regel das Session Description Protocol (SDP) verwendet. Um Beschreibungen für andere Arten von Sitzungen ebenfalls austauschen zu können, ist es möglich, auch andere Beschreibungsprotokolle zu verwenden.

Signalisierung - SIP [47]. SIP ist ein textbasiertes Protokoll, das auf Elementen des Hypertext Transfer Protocol (HTTP) [48] und des Simple Mail Transport Protocol (SMTP) [49] basiert. Die einzelnen SIP-Nachrichten sind in Anfragen (z.B. *INVITE*, *ACK*, *BYE*) und Antworten (z.B. *404 Not Found*) unterteilt. Die SIP-Nachrichten bestehen aus einem Feld, das den Typ der Anfrage bzw. der Antwort beschreibt, gefolgt von einigen - E-Mail ähnlichen - Feldern (z.B. *To*, *From*, *Subject*). Nach diesen Feldern folgt der Body der Nachricht, der beispielsweise eine SDP Sit-

zungsbeschreibung beinhalten kann. Die SIP-Nachrichten werden zwischen den verwendeten SIP-Komponenten ausgetauscht, um eine Mulimedia-Sitzung zu verwalten (siehe Abbildung 15).

Signalisierung - SDP [50]. Das Session Description Protocol (SDP) ist eine reine Formatspezifikation, die festlegt, wie eine Medienbeschreibung aussehen muss. Mit Hilfe von SDP werden Informationen zwischen Sender und Empfänger über die Medien ausgetauscht, die diese zu deren Verarbeitung benötigen. Dazu gehören unter anderem die Formate, verwendete Kodierungsverfahren und Adressen. SDP-Nachrichten werden in diesem Kontext innerhalb von SIP-Nachrichten übermittelt.

Medien - RTP/RTCP [41]. Der Medientransport erfolgt in der Regel wie bei H.323 über RTP/RTCP. Siehe dazu Seite 28. Es können aber auch andere Protokolle und Mechanismen für den Transport der Medien verwendet werden, wenn dies notwendig ist.

Komponenten. Innerhalb eines SIP-Szenarios werden in der Regel folgende Komponenten verwendet:

- **User Agent:** Der User Agent stellt das Interface zum Benutzer dar. Ein User Agent in einem SIP-Telefonieszenario ist dementsprechend ein IP-Telefon.
- **SIP-Server:** In einem SIP-Szenario können verschiedene SIP-Server verwendet werden, die sich durch die bereitgestellten Funktionen unterscheiden (z.B. Registrar, Proxy). Ein User Agent sendet Nachrichten an den entsprechenden SIP-Server, um die angebotene Funktionalität zu nutzen. Ein Proxy-Server wird verwendet, um SIP-Nachrichten an einen anderen SIP-Server weiterzuleiten, der den angeforderten Dienst (z.B. Voice-Mail, Conferencing) bereitstellt. Ein Registrar wird verwendet, um Teilnehmer durch eine Registrierung innerhalb eines SIP-Szenarios bekannt zu machen.
- **Gateway:** Gateways werden verwendet, um einem User Agent die Kommunikation mit Teilnehmern zu ermöglichen, die sich nicht direkt innerhalb des SIP-Szenarios befinden. In einem SIP-Telefonieszenario sind beispielsweise Übergänge in das PSTN-Netz oder in ein H.323-Szenario denkbar.

SIP-Szenario. In Abbildung 14 ist ein SIP-Szenario dargestellt. Verdeutlicht wird die Kommunikation zwischen zwei SIP-Domänen über das dazwischen liegende Internet.

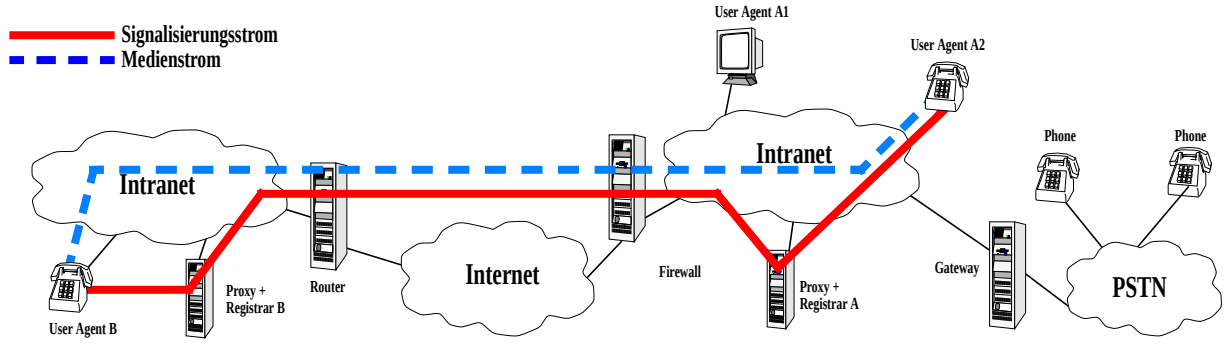


Abbildung 14: SIP-Szenario

User Agent A (UA A) ist bei Registrar A registriert, User Agent B (UA B) bei Registrar B. Beide Registrar-Server besitzen zusätzliche Proxy-Funktionalität. Wird das Gespräch durch UA A initiiert, ergibt sich im Wesentlichen folgender Ablauf: Der UA A sendet eine *INVITE*-Nachricht an seinen Proxy/Registrar A, dieser leitet die Anfrage an Proxy/Registrar B weiter, der dann die Nachricht an UA B weiterleitet. Dieser antwortet mit einer *200 OK*-Nachricht, die in diesem Beispiel den selben Weg nimmt wie die *INVITE*-Nachricht. Die *200 OK*-Nachricht wird nun durch UA A mit einer *ACK*-Nachricht bestätigt. Damit ist die Verbindung zwischen UA A und UA B aufgebaut. Die Beschreibung der Medienströme wird mit Hilfe einer SDP-Beschreibung innerhalb des Bodies der *INVITE*-, *200 OK*- und *ACK*-Nachricht festgelegt.

3.5.2 Angriffe auf SIP-Systeme

Innerhalb eines SIP-Szenarios können Angriffe auf die Signalisierungsebene, Medientransportebene sowie die Verwaltungsebene durchgeführt werden. Diese Möglichkeiten werden nachfolgend genauer betrachtet, allerdings werden mögliche Angriffe - analog zu den H.323-Untersuchungen - auf nicht SIP-spezifische Angriffspunkte (z.B. die Betriebssysteme) nicht betrachtet.

Angriffsmöglichkeiten. Mögliche Angriffe sind im Folgenden kurz dargestellt, weiter Angriffsszenarien und Möglichkeiten sind in [51] und [52] ausführlich dargestellt.

- **Manipulation der Signalisierung:** Einem Angreifer ist es möglich, SIP-Nachrichten zu lesen, zu manipulieren oder aber nicht in den Kontext der Kommunikation passende Nachrichten zu generieren. Es sind dadurch folgende Angriffe denkbar:
  DoS-Angriffe: Durch das Generieren fehlerhafter oder einer großen Anzahl Nachrichten kann es möglich sein, bestimmte SIP-Komponenten außer Funktion zu setzen.
  Unberechtigte Nutzung: Es kann einem Angreifer möglich sein, einen Dienst innerhalb eines SIP-Szenarios unberechtigt zu nutzen (z.B. ein Gateway in das PSTN-Netz).
  Ausspionieren der Netzwerkstruktur: Alle SIP-Nachrichten enthalten Informationen, die einem Angreifer Aufschluss über die interne Struktur des Netzes geben. Beispielsweise wird die Call-ID, die in jeder SIP-Nachricht vorhanden ist, mit Hilfe der IP-Adresse des User Agents generiert. Selbst in einem NAT-Szenario können so interne IP-Adressen durch den Angreifer ermittelt werden.
- **Angriff auf die administrative Schnittstelle:** Besitzt eine RTSP-Komponente eine administrative Schnittstelle, so kann diese ebenfalls für einen Angriff genutzt werden (analog zu Abschnitt 3.4.2).
- **Angriff auf die Vertraulichkeit der Medienströme:** Die Medienströme werden in der Regel nicht verschlüsselt übertragen, ein Abhören ist daher generell möglich.

Maßnahmen. Grundsätzlich können Sicherungsfunktionen auf Applikationsebene für Signalisierungs-, Medien- und Transportebene verwendet werden, wie sie in [47] und [53] vorgeschlagen werden. In diesem Fall muss die Firewall in dieses Sicherungskonzept einbezogen werden. Betrachtet man diese Möglichkeit nicht und geht von einer ungesicherten SIP-Kommunikation aus, so muss die Firewall die SIP-Nachrichten

- auf Korrektheit hinsichtlich Syntax und Kontext prüfen,
- prüfen, ob die Nachrichten vor dem Weiterleiten verändert werden müssen,
- prüfen, ob bestimmte Nachrichten überhaupt weitergeleitet werden sollen,

um die oben dargestellten Bedrohungen zu neutralisieren.

3.5.3 SIP-Applikationen und Firewalls

Die in Abschnitt 3.2.2 beschriebenen Charakteristika einer Multimedia-Applikation existieren ebenfalls innerhalb eines SIP-Szenarios, dadurch besitzen die daraus abgeleiteten Designprinzipien (Abschnitt 3.3.2) auch in einem SIP-Szenario Gültigkeit. Im Folgenden wird die Beschreibung einer SIP-Kommunikation in einem einfachen Szenario gegeben. Davon ausgehend werden dann die einzelnen Charakteristika dieser Applikation dargestellt. In dem Folgenden Beispiel wird angenommen, dass UA A eine bidirektionale Audioverbindung (entsprechend einem Telefongespräch) mit UA B ohne die Nutzung von Infrastrukturkomponenten aufbauen möchte.

- **SIP-Sitzungsaufbau (UDP):** Um die Verbindung aufzubauen, sendet UA A eine *INVITE*-Nachricht an UA B. Diese Nachricht wird in der Regel über UDP verschickt, der SIP-Standard ermöglicht es aber auch, TCP zu verwenden. Die UDP-Nachricht wird an den Well Known Port 5060 geschickt. UA B antwortet mit einer *200 OK*-Nachricht, diese wird dann nochmals durch UA A durch eine *ACK*-Nachricht bestätigt.
  Die verwendeten Eigenschaften der Medienströme können innerhalb der SIP-Nachrichten *INVITE* und *200 OK* durch SDP-Beschreibungen ausgehandelt werden. Dies beinhaltet beispielsweise die für die Medien verwendeten Portnummern.
- **RTP/RTCP Media und Mediacontrol (UDP):** Nachdem die Sitzung über den Austausch der SIP-Nachrichten aufgebaut wurde, können zwischen den User Agents die RTP/RTCP-Medienströme verwendet werden.

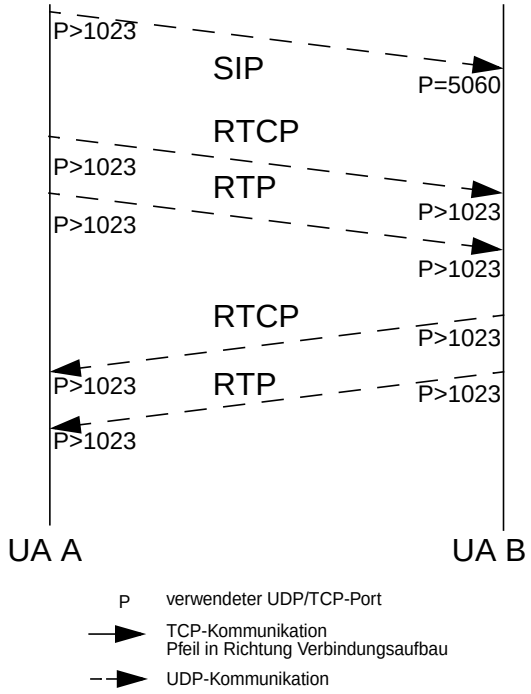


Abbildung 15: Ablauf einer SIP-Sitzung

Zum Beenden der Sitzung wird beispielsweise durch UA A eine *BYE*-Nachricht an UA B geschickt. Diese wird von UA B durch eine *ACK*-Nachricht bestätigt. Danach ist die Sitzung beendet.

Protokollcharakteristika - Komplexität der Protokolle. Wie beschrieben, verwenden SIP-Nachrichten zwei unterschiedliche Blöcke: Den SIP-spezifischen Teil der Nachricht, sowie die im Body der Nachricht transportierten Medienbeschreibungen (idR. SDP-Beschreibungen). Zusätzlich werden für die Übertragung der Medien weitere spezialisierte Protokolle verwendet (idR. RTP/RTCP). Für alle verwendeten Protokolle müssen eigene Protokollautomaten bzw. Parser verwendet werden, die untereinander Abhängigkeiten besitzen. Die SIP-Nachrichten sowie die SDP-Beschreibungen werden in ASCII-Form übermittelt. Dies erfordert innerhalb der SIP-Komponenten einen relativ komplexen Parser, der in der Lage ist, alle Variationen gültiger Nachrichten zu dekodieren.

Protokollcharakteristika - Mehrere Ströme in einer Sitzung. Für eine Sitzung werden mehrere Ströme benötigt. Neben dem UDP-Strom, der für die Signalisierungsnachrichten verwendet wird, werden sperate Ströme für den Transport der Medien benutzt.

Protokollcharakteristika - Dynamisches Verhalten. Neben den dynamisch vereinbarten Parametern (z.B. die verwendeten Portnummern) der Medienströme können während der Laufzeit einer Sitzung neue Medienströme hinzukommen und andere beendet werden.

Applikationscharakteristika - Szenarienvielfalt und Szenarienkomplexität. Die Kommunikationsabläufe innerhalb eines SIP-Szenarios hängen von der Beschaffenheit des verwendeten Szenarios ab. Je nach verwendeten Infrastrukturkomponenten und deren Konfiguration werden zum Aufbau einer Sitzung verschiedenartige Kommunikationsabläufe benötigt.

Applikationscharakteristika - Interpretation der Protokollstandards. Der SIP-Standard lässt - Ähnlich dem H.323-Protokoll - verschiedene Varianten zu, um ein und die selbe Aufgabe auszuführen. Beispielsweise ist es möglich innerhalb einer INVITE-Nachricht keine Medienbeschreibung zu schicken; in diesem Fall werden dann die Medienbeschreibungen in der *200 OK* und der darauffolgenden *ACK*-Nachricht festgelegt (vergl. Variante oben).

Applikationscharakteristika - Verwendung zusätzlicher Protokolle. Es können ebenfalls zusätzliche Protokolle verwendet werden. Beispielsweise kann das DNS-Protokoll verwendet werden, um ein Call-Routing durchzuführen (siehe [54]).

Leistungscharakteristika. Die Leistungsanforderungen entsprechen im Wesentlichen denen einer H.323-Applikation, wenn davon ausgegangen wird, das ein SIP-Szenario zur Umsetzung einer IP-Telefonie Lösung verwendet wird. Sollen andere Szenarien bedient werden (z.B. interaktive Spiele) muss jeweils betrachtet werden, welche Bedingungen dafür eingehalten werden müssen.

3.6 RTSP-Applikationen

Das Real Time Streaming Protocol (RTSP) wird für die Steuerung der Übertragung von Daten mit Echtzeitanforderungen verwendet. Bei den zu transportierenden Daten handelt es sich in der Regel um Audio- oder Videodaten. Im Gegensatz zu den bereits beschriebenen IP-Telefonie Protokollen H.323 und SIP wird RTSP vorwiegend für reine Client-Server Anwendungen verwendet. Der Client fordert über das RTSP-Protokoll ein bestimmtes Medium (z.B einen Film oder eine Audioquelle) bei einem Server an, dieses wird dann zum Client übertragen und dort dargestellt. RTSP wird zur Zeit häufig für die Realisierung von Video on Demand Diensten [55] oder Internet-Radio Anwendungen [56] verwendet.

Im Folgenden werden die wesentlichen Aspekte des Zusammenspiels von RTSP-Applikationen mit Firewalls dargestellt. Eine ausführliche Beschreibung dieser Sachverhalte findet sich in [57].

3.6.1 Grundlagen

RTSP wurde im April 1998 als Proposed Standard RFC2326 durch die IETF verabschiedet. Der RTSP-Standard beschreibt die notwendige, zwischen Client und Server ablaufende Signalisierung, die für die Medienübertragung zwischen den Endpunkten nötig ist. Auf welche Art und in welcher Form die Medien übertragen werden wird über die RTSP-Signalisierung zwischen den Endpunkten festgelegt. Die Form der dazu verwendeten Präsentationsbeschreibungen ist nicht innerhalb des RTSP-Standards festgelegt, typischerweise wird hier aber das Session Description Protocol (SDP) verwendet.

Signalisierung - RTSP [58]. Der RTSP-Kanal wird für den Aufbau, Abbau und die Kontrolle einer Sitzung zwischen Client und Server verwendet. Dazu werden entsprechende RTSP-Nachrichten zwischen Client und Server ausgetauscht. Die RTSP-Nachrichten sind textbasierte Nachrichten mit einem Header und einem optionalen Body und bauen auf HTTP auf. Es wird zwischen Nachrichten vom Typ “Anfragen” und “Antworten” unterschieden. Eine RTSP-Anfrage besteht aus dem Namen der RTSP-Methode (z.B. *DESCRIBE*, *SETUP*) der URI für die angeforderte Resource, der RTSP-Version und verschiedenen Headerzeilen. Eine RTSP-Antwort besteht aus der RTSP-Version, einem Statuscode, einer textuellen Beschreibung des Statuscodes sowie ebenfalls verschiedener Headerzeilen. In diesen Headerzeilen stehen beispielsweise Bestätigungen für eine Anfrage. Sollte eine RTSP-Nachricht einen Body besitzen, dann enthält der Header eine Angabe über Länge und Typ des Bodies. Der Body kann zum Beispiel eine SDP-Medienbeschreibung beinhalten. Der Transport der RTSP-Nachrichten erfolgt in der Regel über TCP, der Standard erlaubt aber auch UDP als Transportprotokoll.

Signalisierung - SDP [50]. Das Session Description Protocol (SDP) wird auch innerhalb von SIP verwendet und wurde bereits dort kurz beschrieben. Die SDP-Nachrichten werden in diesem Kontext innerhalb von RTSP-Nachrichten übermittelt.

Medien - RTP/RTCP [41]. Der Medientransport erfolgt wie bei H.323 und SIP in der Regel über RTP/RTCP. Siehe dazu Seite 28. Der Standard erlaubt aber auch die Verwendung anderer Protokolle.

RTSP-Szenario. In Abbildung 16 ist ein Video on Demand (VoD) Szenario basierend auf dem RTSP-Protokoll dargestellt. Es existiert ein Origin-Server im Internet, der die durch die VoD-Struktur zu verteilenden Filme vorhält.

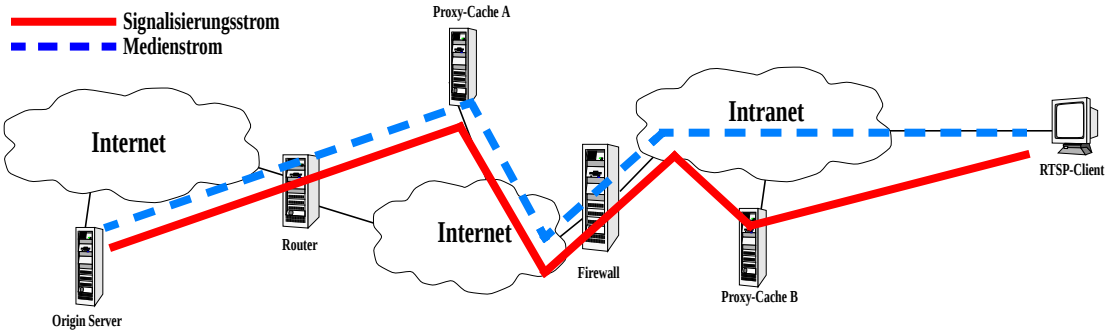


Abbildung 16: RTSP-Szenario

Ein Client, der nun einen Film anfordern möchte, wendet sich zuerst an seinen lokalen Proxy, um, falls möglich eine zwischengespeicherte (Cache) Variante direkt von dort zu beziehen. Ist diese nicht vorrätig, wird der Film durch den Proxy von dem nächst höheren Proxy innerhalb der Hierarchie angefordert. Ist der Film auch dort nicht vorrätig, wird der Origin-Server als höchste Instanz in der Hirarchie befragt. Im obigen Beispiel wird der Film über den Proxy Cache A an den Client ausgeliefert. Cache A hat in diesem Fall die Möglichkeit den Film für spätere Anfragen zwischenzulagern. Cache B wird in diesem Fall von den Medien nicht durchlaufen. Dies kann beispielsweise dann geschehen, wenn festgelegt wurde, dass Cache B diese Möglichkeit nicht wahrnehmen soll.

3.6.2 Angriffe auf RTSP-Systeme

Es können Angriffe auf die Signalisierungsebene, Medientransportebene sowie die Verwaltungsebene eines RTSP-Systems durchgeführt werden. Diese Möglichkeiten werden nachfolgend genauer betrachtet. Darüber hinaus existieren analog zu den H.323-Untersuchungen (siehe Abschnitt 3.4.2) auch weitere, nicht RTSP spezifische Angriffspunkte (z.B. die Betriebssysteme welche die RTSP-Komponenten tragen), welche im Folgenden allerdings nicht betrachtet werden.

Angriffsmöglichkeiten. Es wurde untersucht, welche Angriffe innerhalb eines RTSP-Systems theoretisch möglich sind. Eine detaillierte Beschreibung dieser Angriffsmöglichkeiten ist in [57] gegeben, nachfolgend wird eine Zusammenfassung dieser gegeben.

- **Manipulation der Signalisierung:** Das Senden von modifizierten oder nicht in den Kontext der aktuellen Kommunikation passenden Signalisierungsnachrichten an eine RTSP-Komponente (Server oder Client) kann zu einer Fehlfunktion der Komponente führen (DoS). Eventuell kann ein Angreifer durch ein solches Vorgehen auch eine Schwachstelle innerhalb der Komponente für seine Zwecke nutzen. Folgende Angriffe

sind denkbar:

Manipulation der Konfiguration: Mit den Methoden *GET_PARAMETER* und *SET_PARAMETER* können Client und Server Parameter austauschen und ändern. Diese Parameter sind implementierungsspezifisch und es ist denkbar, dass dadurch auch sicherheitsrelevante Parameter veränderbar sind.

Ablegen von eigenen Daten: RTSP unterstützt die Methode *RECORD*, die es einem Client ermöglicht Medien auf einen Server zu spielen. Wird diese Methode von einem Server unterstützt, bietet sie einem Angreifer die Möglichkeit Ressourcen des Servers zu belegen (z.B. Plattenplatz, CPU, IO). Neben dem unerwünschten Auftreten von Daten innerhalb des Servers kann dies auch zu einem DoS führen.

Ausspionieren der Netzwerkstruktur: Manche in RTSP-Nachrichten verwendete Header geben Aufschluss über die Beschaffenheit des Netzes und können von einem Angreifer zur Vorbereitung eines Angriffs verwendet werden. Zum Beispiel beschreibt der *Server*-Header den Typ des verwendeten Servers. Ist einem Angreifer bekannt, dass dieser Typ eine Schwachstelle besitzt ist dann ein gezielter Angriff auf diese Schwachstelle möglich.

Missbrauch des Servers: Ein Client kann von einem Server ein bestimmtes Medium anfordern. In der Beschreibung der Zieladresse des Medienstroms kann der Client eine andere Adresse als seine eigene angeben. Dadurch erhält dieser andere Rechner nun den Medienstrom. Ein Client kann so einen Server für eine DoS-Attacke auf andere Rechner nutzen. Der Standard sieht deshalb vor, dass diese Funktion nur nach Authentifizierung des Clients verwendet werden darf. Dennoch halten sich nicht alle Server an diese Vorgabe.

- **Angriff auf die administrative Schnittstelle:** Besitzt eine RTSP-Komponente eine administrative Schnittstelle, so kann diese ebenfalls für einen Angriff genutzt werden (analog zu Abschnitt 3.4.2).
- **Angriff auf die Vertraulichkeit der Medienströme:** Die Medienströme werden in der Regel nicht verschlüsselt übertragen, ein Abhören ist daher generell möglich.

Maßnahmen. Wird die Kommunikation zwischen Angreifer und Ziel über eine Firewall hinweg abgewickelt, so können die meisten der oben beschriebenen Bedrohungen neutralisiert werden. Die im ersten Punkt zusammengefassten Angriffe können durch Kontrollen der Signalisierungsnachrichten verhindert werden. Es muss geprüft werden, ob die gesendeten Nachrichten

- eine korrekte Struktur besitzen, um zu verhindern, dass irregulären RTSP-Nachrichten weitergeleitet werden.
- in den Kontext der aktuell stattfindenden Kommunikation passen, um unerwartete PDU-Abfolgen zu verhindern.
- überhaupt weitergeleitet werden sollen, um das Auslösen bestimmter Funktionen über die Netzgrenze hinweg zu verhindern.

- vor einem Weiterleiten verändert werden müssen, um bestimmte sicherheitskritische Elemente der Nachrichten zu entfernen.

Die im zweiten Punkt beschriebenen Angriffe können durch eine Firewall verhindert werden, indem ein administrativer Zugriff über die Netzgrenze hinweg durch eine Firewall unterbunden wird.

Wie auch in den vorhergehenden Beispielen ergibt sich, dass eine Überprüfung der Signalisierung durch eine Firewall auf Applikationsebene notwendig ist. Eine Überprüfung der Medienströme auf Applikationsebene ist nicht notwendig. Zumindest die hier dargestellten Angriffe können dadurch nicht unterbunden werden. Die Untersuchung zeigt, dass die in Abschnitt 3.3.1 getroffenen Annahmen über die auszuführenden Sicherheitsüberprüfungen der Signalisierungs- und Medienströme zutreffen.

3.6.3 RTSP-Applikationen und Firewalls

Die in Abschnitt 3.2.2 beschriebenen Charakteristika von Multimedia-Applikationen sind auch innerhalb eines RTSP-Szenarios zu finden. Damit besitzen auch die in Abschnitt 3.3.2 davon abgeleiteten Designprinzipien Gültigkeit innerhalb eines RTSP-Szenarios. Die einzelnen Charakteristika sowie die dadurch verursachten Probleme werden im Folgenden beleuchtet.

Zunächst wird als Basis der folgenden Beschreibungen eine einfache RTSP-Kommunikation zwischen Client und Server beschrieben. In diesem Beispiel wird durch den Client das Abspielen eines Films von einem Videoserver gestartet.

- **RTSP Sitzungsaufbau (TCP):** Der Client öffnet einen TCP-Kanal, dazu wird Port 554 (der RTSP Well Known Port) verwendet. Dieser Kanal wird nun zwischen Client und Server für den Austausch der RTSP-Nachrichten verwendet.
  Der Client fragt den Server nach einer Beschreibung für eine Videoresource (*DESCRIBE*). Der Server antwortet mit *OK* und schickt im Body eine SDP-Beschreibung des Videos. Der Client initiiert daraufhin eine neue Sitzung und gibt dazu in der Transportzeile der *SETUP*-Nachricht, die von ihm akzeptierten Transportparameter (Port, IP-Adresse und Transportprotokoll des folgenden RTP-Stroms) an. Der Server antwortet mit *OK*. Der Client startet das Abspielen, indem er eine *PLAY*-Nachricht an den Server sendet. Der Server antwortet mit einer *OK*-Nachricht und startet die Medienübertragung.
- **RTP/RTCP Media und Mediacontrol (UDP):** Nach dem oben beschriebenen Ablauf startet der

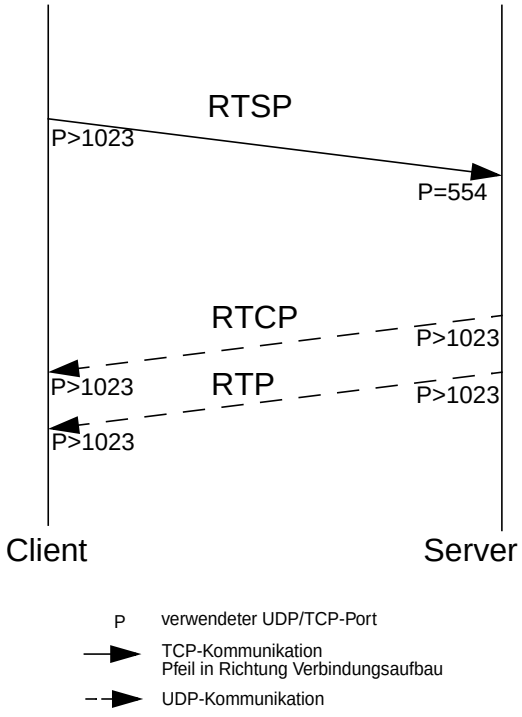


Abbildung 17: Ablauf einer RTSP-Sitzung

Server das Abspielen des Films. Der RTP- und RTCP-Strom wird initiiert unter der Verwendung der durch den Client übermittelten Transportspezifikationen.

Zum Beenden der Sitzung wird durch den Client eine *TEARDOWN*-Nachricht an den Server gesendet, diese wird durch ein *OK* vom Server bestätigt. Danach wird der RTP/RTCP-Strom gestoppt und dann die RTSP-Kontrollverbindung terminiert.

Protokollcharakteristika - Komplexität der Protokolle. Wie beschrieben werden innerhalb der RTSP-Nachrichten auch SDP-Nachrichten übermittelt. Für die Übertragung der Medien wird RTP und RTCP verwendet. Sowohl für RTSP als auch für SDP und RTP/RTCP müssen jeweils eigene Protokollautomaten bzw. Parser innerhalb einer RTSP-Komponente erstellt werden. Diese Komponenten besitzen zusätzlich Abhängigkeiten untereinander. Die einzelnen RTSP-Nachrichten sowie die SDP-Beschreibungen werden in ASCII-Form übermittelt. Dies erfordert innerhalb der RTSP-Komponenten einen komplexen Parser, der in der Lage ist alle Variationen gültiger Nachrichten zu dekodieren.

Protokollcharakteristika - Mehrere Ströme in einer Sitzung. Eine RTSP-Sitzung verwendet den RTSP-Kontrollkanal. Zusätzlich werden für die zu übermittelnden Medien entsprechend viele RTP- und RTCP-Ströme benötigt.

Protokollcharakteristika - Dynamisches Verhalten. Die für die Medien verwendeten Transportcharakteristika (z.B. Protokolle, IP-Adressen und Portnummern) werden dynamisch über den Kontrollkanal zur Laufzeit einer Sitzung ausgehandelt, bzw. festgelegt.

Applikationscharakteristika - Szenarienvielfalt und Szenarienkomplexität. Die verwendeten Kommunikationsmechanismen innerhalb eines RTSP-Szenarios hängen von den innerhalb des Szenarios verwendeten Komponenten, sowie von ihrer jeweiligen Beschaffenheit ab. Wird beispielsweise anstelle der in Abbildung 17 dargestellten direkten Kommunikation ein VoD-Szenario entsprechend Abbildung 16 verwendet, kann sich folgende Änderung des Kommunikationsverhaltens ergeben: Die Signalisierung läuft immer noch zwischen Client und Server (bzw. Proxy Cache), der Medienstrom wird aber nun von einem weiteren Server geliefert.

Applikationscharakteristika - Interpretation der Protokollstandards. Die textbasierten Protokolle RTSP sowie SDP lassen viele Interpretationsmöglichkeiten zu. Dies ist beispielsweise daran zu erkennen, dass verschiedene RTSP-Clienten nicht mit jedem RTSP-Server zusammenarbeiten, obwohl alle Komponenten RTSP-Konform sind [59].

Applikationscharakteristika - Verwendung zusätzlicher Protokolle. Um eine effiziente Kommunikation in VoD-Hierarchien zu erreichen werden oft zusätzliche bzw. modifizierte Protokolle zwischen Origin-Servern und den einzelnen Proxy-Caches verwendet (z. B. LCRTP [60]).

Leistungscharakteristika. Die Leistungsanforderungen entsprechen im Wesentlichen denen der bereits beschriebenen Protokolle. Da die Übertragung der Medien, im Gegensatz zu den bereits beschriebenen Telefonie-Protokollen, unidirektional stattfindet, wirken sich Verzögerungen auf den Sitzungsaufbau - Verzögerungen innerhalb des Kontrollkanals - weniger kritisch aus. Es ist in

der Regel nicht problematisch, wenn nach Client-Anforderung die Medienübertragung mit einer gewissen Verzögerung beginnt. Für die dann folgende Medienübertragung gelten aber die strengen Anforderungen hinsichtlich der Verzögerung, des Jitters und des Paketverlustes.

3.7 Zusammenfassung

In diesem Kapitel wurde dargelegt, welche Probleme bei der Verwendung von Multimedia-Applikationen in durch Firewalls geschützten Netzwerken auftreten. Die Ursachen der Probleme, die charakteristischen Eigenschaften der Multimedia-Applikationen, wurden klassifiziert und es wurden daraus entsprechende Lösungsvorschläge in Form von Designprinzipien entwickelt.

Es wurde gezeigt, dass die Charakteristika und die davon abgeleiteten Designprinzipien zumindest für drei verschiedene Multimedia-Protokolle Gültigkeit besitzen. Damit ist nicht zweifelsfrei bewiesen, dass dies für alle existierenden Multimedia-Protokolle gilt. Dennoch ist abzusehen, dass die getroffenen Aussagen mit hoher Wahrscheinlichkeit auch für andere Multimedia-Protokolle Gültigkeit besitzen. Zumindest ein Großteil der heute auftretenden Multimedia-Kommunikation, die auf den hier beschriebenen Protokollen (H.323, SIP, RTSP) basiert, kann mit den gefundenen Erkenntnissen erfasst werden.

Die folgenden Kapitel beschäftigen sich mit der Umsetzung der in diesem Kapitel vorgestellten Lösungsansätze. Es wird gezeigt, dass diese Lösungsansätze auch zu einer Lösung der beschriebenen Probleme führen.

Kapitel 4: Firewall-Architekturen

In Kapitel 2 wurden die von einer Firewall zu erbringenden Funktionen beschrieben. In Kapitel 3 wurde beschrieben, welchen zusätzlichen Anforderungen eine Multimedia-Firewall gerecht werden muss. Wie gezeigt, ist nicht festgelegt, welche Firewall-Architektur verwendet werden muss, um die notwendigen Firewall-Funktionen zu erbringen, bzw. die Anforderungen zu erfüllen. In diesem Kapitel wird nun untersucht, welche Architekturen sich prinzipiell für die Unterstützung von Multimedia-Applikationen eignen.

Im Folgenden wird ein Architekturmodell vorgestellt, das es ermöglicht, Firewall-Architekturen hinsichtlich der für die Verarbeitung von Multimedia-Applikation relevanten Architekturmerkmale zu ordnen. Das Modell ermöglicht es, Gemeinsamkeiten - bzw. die Verwendung identischer Elemente - innerhalb verschiedener Architekturklassen zu identifizieren. Durch die Überprüfung, welche Architekturklasse sich wie gut für die Unterstützung von Multimedia-Applikationen eignet, kann nach Einordnung bestehender Firewall-Systeme eine generelle Aussage über deren Eignung für Multimedia-Applikationen getroffen werden.

Im Weiteren werden innerhalb dieses Kapitels verschiedene zur Zeit verwendete oder diskutierte Firewall-Architekturen anhand des zuvor definierten Architekturmodells betrachtet und bewertet. Dies ermöglicht es, für die Unterstützung von Multimedia-Applikationen vorhandene sinnvolle, bzw. noch fehlende, Architekturelemente zu identifizieren. In Kapitel 5, Kapitel 6 und Kapitel 7 wird dann genauer auf Entwurf bzw. Verwendung dieser identifizierten Elemente eingegangen.

4.1 Architekturmodell

Damit die einzelnen Komponenten einer Firewall für Multimedia-Applikationen ihre Funktionen erfüllen können, ist ein Informationsaustausch zwischen diesen notwendig. Im wesentlichen sind dabei Informationen über die verwendeten Ströme einer Multimedia-Sitzung von Bedeutung. Das folgende Beispiel soll dies verdeutlichen:

Angenommen die Firewall besteht aus einem Paketfilter, durchlaufen von den Signalisierungs- und Medienströme, sowie einem Proxy, durchlaufen von den Signalisierungsströmen. In diesem Fall benötigt der Paketfilter Informationen über den aktuellen Zustand der Kommunikationskanäle, um seine Konfiguration an den aktuellen Zustand anpassen zu können. Diese Informationen können innerhalb der Firewall nur durch den Proxy bereitgestellt werden, da nur diese Komponente die nötigen Informationen besitzt. Es ist also eine Übergabe der aktuellen Kanalspezifikationen von dem verwendeten Proxy an den Paketfilter notwendig. Entsprechende Schnittstellen zwischen Proxy und Paketfilter müssen dazu vorhanden sein.

Die einzelnen logischen Elemente, die an einem solchen Informationsaustausch beteiligt sind, sind in Abbildung 18 dargestellt.

Abbildung 18: Logische Elemente einer Firewall

Wie in den folgenden Abschnitten dargestellt, kann in der Regel in jeder Multimedia-Firewall dieser für den Informationsaustausch verwendete Pfad identifiziert werden. In Abschnitt 4.4 wird gezeigt, dass sich die verschiedenen möglichen Firewall-Architekturen aus der örtlichen Anordnung und der Art der Umsetzung der einzelnen logischen Elemente ergeben. Es ist ebenfalls zu beachten, dass alle logischen Elemente, unabhängig von ihrer örtlichen Plazierung, Teil des Firewall-Systems sind.

Informationsaustausch. Es ist notwendig die Kanalspezifikationen der einzelnen Ströme von der Applikation zur Firewall (bzw. auf die entsprechenden in der Firewall verwendeten Komponenten) zu transportieren. Die Kanalspezifikation kann in ihrer einfachsten Form durch das in (1) dargestellte 5-Tupel beschrieben werden:

$$F = \{PROTO, SRCIP, SRCPORT, DSTIP, DSTPORT\} \quad (1)$$

Falls es zur Umsetzung der für die Firewall gültigen Security Policy notwendig ist, muss das angegebene 5-Tupel entsprechend erweitert werden (z.B. um eine Angabe, die den Benutzer des entsprechenden Kanals ausweist). Zusätzlich werden Informationen in Gegenrichtung transportiert. Es muss beispielsweise der Applikation, bzw. dem Szenario, in dem die Applikation verwendet wird, mitgeteilt werden, ob das Übermitteln einer Kanalspezifikation erfolgreich war. Eine vollständige Liste aller möglichen auszutauschenden Informationen kann nicht gegeben werden. Menge und Art der auszutauschenden Informationen hängt von der Beschaffenheit der Applikation und der Firewall ab.

Application. Dieses Element beschreibt die Applikationsseite. Es ist damit nicht unbedingt die Applikation selbst gemeint, sondern vielmehr das Szenario, in dem die Applikation verwendet wird.

Application Control. Das als *Application Control* bezeichnete Element ist in der Lage, die für die Firewall-Seite notwendigen Informationen von der Applikationsseite zu beschaffen. Gleichzeitig kann dieses Element von der Firewall-Seite gelieferte Informationen der Applikationsseite mitteilen, bzw. die Verwendung dieser Informationen durchzusetzen. Dieses Element stellt damit die Schnittstelle zwischen Multimedia-Applikation und Firewall-System dar.

Die technische Realisierung kann auf verschiedene Arten erfolgen. Es ist möglich, die *Application Control* innerhalb eines Endgerätes (z.B. eines H.323-Terminals) unterzubringen. Die Kanalspezifikationen werden in diesem Fall von einem Endgerät an die Firewall übergeben. Eine andere Methode ist die Realisierung der *Application Control* durch einen im Signalisierungsweg liegenden Proxy (siehe Kapitel 6). In diesem Fall werden die Kanalspezifikationen aus der ablaufenden Kommunikation extrahiert und an die Firewall weitergegeben. Es sind weitere technische Realisierungen dieser Einheit möglich, die aber dem gleichen Zweck dienen.

Signalling Part. Der *Signalling Part* wird dazu verwendet die von der *Application Control* an die *Firewall Control* zu übergebenden Informationen zu transportieren. Befinden sich die technischen Umsetzungen von *Firewall Control* und *Application Control* innerhalb der gleichen Komponente, so besteht der *Signalling Part* aus einer einfachen API. Sind die durch den *Signalling Part* zu verbindenden logischen Einheiten auf unterschiedlichen Komponenten untergebracht, so beinhaltet der *Signalling Part* ein Netzwerkprotokoll, um die Informationen über das zwischen den Komponenten liegende Netzwerk auszutauschen. Dieses Netzwerkprotokoll wird innerhalb dieser Arbeit als Firewall Control Protocol (FCP) bezeichnet.

Firewall Control. Dieses Element ist dafür verantwortlich die - entsprechend der übermittelten Informationen - notwendigen Konfigurationsanpassungen innerhalb der Firewall-Komponenten vorzunehmen. Gleichzeitig werden die für die Applikationsseite bestimmten Informationen von der Firewall-Seite entgegengenommen. Es muss durch dieses Element ebenfalls festgelegt werden, nach welchen Regeln die übermittelten Informationen an die Firewall-Komponenten übergeben werden (Festlegen einer Policy). Beispielsweise kann es gewünscht sein, keine Kanalspezifikationen an die Komponenten weiterzuleiten, die eine Port-Nummer kleiner 1024 für den SRC-Port oder DST-Port angeben. Außerdem muss hier die Umwandlung der übermittelten generischen Kanalspezifikationen in die jeweiligen, für Firewall-Komponenten spezifischen Formate, durchgeführt werden. Besteht die Firewall-Seite aus mehreren Firewall-Komponenten, so muss eine Unterverteilung der Information auf die verschiedenen Firewall-Komponenten erfolgen. Wie dies effizient realisiert werden kann, ist in [31] beschrieben.

Firewall. Dieses Element beschreibt die Firewall-Seite. Die Firewall-Seite kann wiederum selbst aus mehreren Firewall-Komponenten bestehen.

4.2 Architekturklassen

Im Folgenden werden verschiedene Firewall-Architekturen dargestellt, sowie ihre Vor- und Nachteile aufgezeigt. Die hier gezeigten Architekturen ergeben sich durch die entsprechende örtliche Anordnung der zuvor beschriebenen logischen Elemente des Architekturmodells. Die weiterhin definierten Architekturklassen ermöglichen die Einordnung und damit Bewertung der wesentlichen zur Zeit in der Literatur diskutierten Lösungsansätze. Zur Darstellung werden die folgenden Symbole verwendet:

Abbildung 19: Symbole zur Beschreibung der Firewall-Architekturen

Es wird betrachtet, wie geeignet die einzelnen Architekturklassen für das Zusammenspiel mit Multimedia-Applikationen sind, indem geprüft wird, inwieweit die in Kapitel 3 beschriebenen Designprinzipen umgesetzt werden können.

Architektur I. - Proxy. Ein Proxy besteht aus nur einer einzigen Firewall-Komponente, für deren Umsetzung nur eine einzige physikalische Komponente verwendet wird. Sowohl die Signalisierungs- , als auch die Medienströme werden durch die Proxy-Komponente verarbeitet. In diesem Fall besteht die *Application Control* aus einem Parser innerhalb des Proxies, der die Signalisierungsströme des Multimedia-Protokolls verarbeiten und analysieren kann. Dem entsprechend wird für die Unterstützung verschiedener Multimedia-Protokolle jeweils ein eigenständiger Proxy benötigt. Der in Abbildung 18 beschriebene Informationsfluss findet nur innerhalb der Proxy-Komponente selbst statt. Die durch den Parser gewonnenen Informationen werden direkt durch den Proxy für die Bearbeitung der Medienströme genutzt. Dadurch, dass sowohl Signalisierung als auch Medienströme durch den Proxy verarbeitet werden, wird damit auch automatisch eine NAT-Funktionalität bereitgestellt. Betrachtet man den Proxy unter dem Gesichtspunkt des in Kapitel 3 beschriebenen Designprinzips, so ist folgendes festzustellen:

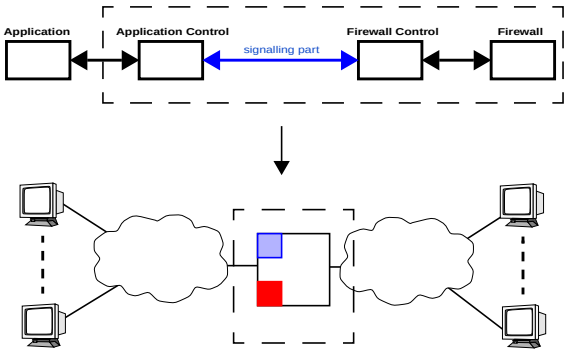


Abbildung 20: Proxy

- **Trennung von Signalisierungs- und Medienpfad:** Eine getrennte Bearbeitung von Signalisierungs- und Medienverarbeitung ist innerhalb eines Proxies nicht möglich. Die damit verbundenen Optimierungen können nicht durchgeführt werden.

Es zeigt sich damit, dass ein Proxy für den Einsatz mit Multimedia-Applikationen nicht geeignet ist. Dies wird insbesondere bei der Betrachtung der Leistungsfähigkeit dieser Systeme (Siehe dazu Kapitel 7) klar. In der Praxis werden Proxies dennoch häufig eingesetzt, da sie den Vorteil

besitzen, dass Implementierung und Wartung einfach durchzuführen sind. Bekannte Proxies sind beispielsweise für H.323-Applikationen der OpenH323proxy [61] und für RTSP Applikationen das RTSP Proxy Kit [62].

Architektur II. - Hybridsystem. Bei einem Hybridsystem werden verschiedene Firewall-Komponenten innerhalb einer physikalischen Komponente realisiert. Die *Application Control* ist in diesem Fall als Parser innerhalb eines Proxies für die Signalisierungsströme ausgeführt. Für jedes durch die Firewall zu unterstützendes Protokoll ist ein spezialisierter Protokoll-Parser notwendig. Die an die *Firewall Control* übermittelten Informationen werden verwendet, um die im Hybrid-System verwendeten restlichen Firewall-Komponenten (z.B. Stateful Filter, NAT-Komponente) an die Kommunikation anzupassen. Der *Signalling Part* kann hier durch relativ simple Funktionsaufrufe umgesetzt werden. Das in Kapitel 3 beschriebene Designprinzip kann folgendermaßen realisiert werden:

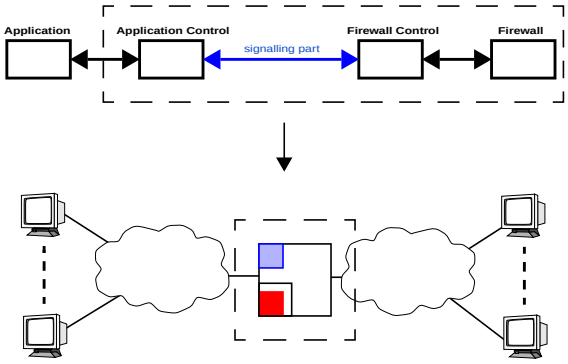


Abbildung 21: Hybridsystem

- **Trennung von Signalisierungs- und Medienpfad:** Eine Trennung der Pfade ist innerhalb des Hybridsystems möglich. Eine Trennung, die auch die Verwendung unterschiedlicher physikalischer Komponenten für beide Pfade vorsieht, ist jedoch nicht möglich. Alle Komponenten teilen sich die innerhalb des Systems verfügbaren Ressourcen (z.B. CPU, Speicher). Eine Optimierung der einzelnen Pfade ist nur begrenzt möglich, da alle Komponenten innerhalb eines Systems umgesetzt werden müssen.

Es zeigt sich, dass ein Hybridsystem zwar besser geeignet ist als ein Proxy, aber dennoch Beschränkungen existieren. Zum einen verfügen heute existierende Hybridsystem nicht über die notwendigen Möglichkeiten, Multimedia-Applikationen sinnvoll zu unterstützen (Siehe Abschnitt 4.4.2). Zum anderen besitzen solche Systeme Beschränkungen in der Gesamtleistung (siehe Kapitel 7). Dennoch wird die Klasse der Hybridsysteme heute in der Praxis am häufigsten eingesetzt, Beispiele hierfür sind [23] und [24].

Architektur III. - Verteiltes System. Das in Abbildung 22 dargestellte verteilte System besteht aus einem Signalisierungsproxy sowie einem Filter inklusive NAT-Komponente. Der Proxy ist mit dem Filter über ein Netzwerk (DMZ) verbunden. Die *Application Control* ist als Parser innerhalb des verwendeten Proxies ausgeführt, der die Kontrollströme des Multimedia-Protokolls verarbeiten und analysieren kann. Die zwischen den Firewall-Komponenten ausgetauschten Informationen müssen bei der hier dargestellten Architektur über ein Netzwerk transportiert werden. Das bedeutet, dass der *Signalling Part* aus einem Netzwerkprotokoll besteht. Das in Kapitel 3 beschriebenen Designprinzip kann folgendermaßen umgesetzt werden:

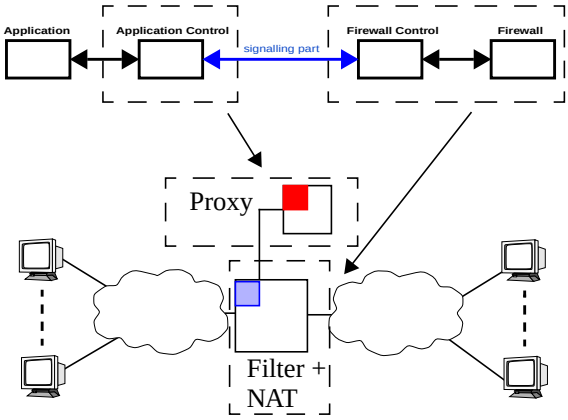


Abbildung 22: Verteiltes System

- **Trennung von Signalisierungs- und Medienpfad:** Eine Trennung von Signalisierungs- und Medienpfad ist möglich. Die Verarbeitung beider Pfade kann dabei durch eigenständige spezialisierte Komponenten übernommen werden. Die Verarbeitungslogik für den Signalisierungs- und den Medienpfad kann getrennt voneinander entworfen werden.

Ein verteiltes System eignet sich besonders für die Unterstützung von Multimedia-Applikationen. Ein in der Praxis weiterer wichtiger Vorteil dieser Architektur besteht darin, dass die Signalisierungsproxies durch bereits existierende, die Signalisierung bearbeitende Infrastrukturkomponenten ersetzt werden können. In einem solchen Fall muss lediglich die *Application Control* in die existierende Komponente integriert werden. In einem H.323-Szenario ist es beispielsweise möglich, den Proxy durch einen H.323-Gatekeeper zu ersetzen. Der H.323-Proxy (bzw. Gatekeeper) enthält dann die *Application Control* sowie die Möglichkeit mit der Filterkomponente zu interagieren. Eine weitere Variante dieser Architektur ist, die Proxies nicht in einem eigenen Netz anzusiedeln, sondern in die zur Kommunikation verwendeten Netze zu integrieren. Beispiele für die hier beschriebene Systemklasse sind in [63] gegeben.

Architektur IV. - Endsystembasierte Firewall. In diesem Fall ist die *Application Control* innerhalb der Endsysteme angesiedelt. Das für die Kommunikation verwendete Netzwerk wird in diesem Fall auch für den Transport der Informationen zwischen *Application Control* und *Firewall Control* verwendet. Die *Application Control* lässt sich innerhalb der Endsysteme relativ einfach realisieren, da dort die notwendigen Informationen (z.B. Kanalspezifikationen) direkt zu extrahieren sind. Verglichen mit den zuvor beschriebenen Architekturklassen müs-

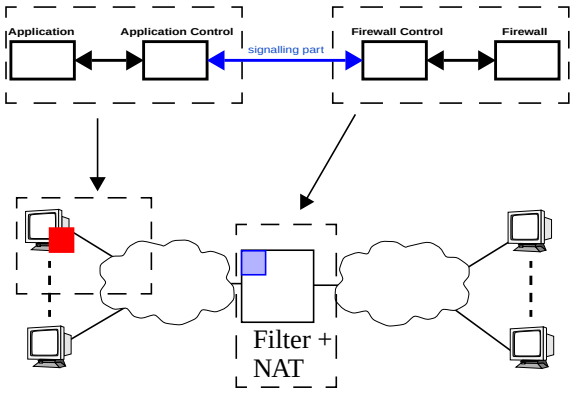


Abbildung 23: Endsystembasierte Firewall

sen die Kanalspezifikationen nicht aus der ablaufenden Kommunikation gewonnen werden, sondern können direkt über einen Funktionsaufruf von der Applikation an die *Application Control* übergeben werden. Allerdings benötigt jedes Endsystem die entsprechende Modifikation, um mit der *Firewall Control* interagieren zu können. Die in Kapitel 3 beschriebenen Designprinzipien können folgendermaßen umgesetzt werden:

- **Trennung von Signalisierungs- und Medienpfad:** Eine Trennung von Signalisierungs- und Medienpfad ist möglich. Die Verarbeitung beider Pfade kann dabei durch spezialisierte Komponenten übernommen werden.
- **Aktive Unterstützung der Firewall:** Die Integration der *Application Control* direkt in die Applikation sorgt für eine aktive Unterstützung der Firewall. Es wird zwar nicht das verwendete Multimedia-Protokoll verändert, es erfolgt aber eine Anpassung der einzelnen Applikationen.

Da es relativ schwierig ist, eine bestehende Infrastruktur zu ändern (Eingriffe in jedem Endgerät sind notwendig), wird diese Architekturklasse in der Praxis nicht häufig eingesetzt. Ein entscheidender Nachteil dieser Architekturklasse ist die hohe Komplexität (durch den hohen Grad der Verteilung) des Firewall-Systems, was unter dem Gesichtspunkt der Sicherheit problematisch ist. Firewall-Systeme, die auf dieser Architekturklasse beruhen sind beispielsweise SOCKS- [64] und TARP-basierte [65] Systeme. Auf die Funktionsweise dieser Lösungen wird in Abschnitt 4.4.4 genauer eingegangen.

4.3 Vergleich der Architekturklassen

Die heute verwendeten und diskutierten Architekturen lassen sich im Wesentlichen auf eine der zuvor beschriebenen Architekturen abbilden. Ein grundsätzlicher Vergleich dieser vier generellen Architekturen hinsichtlich ihrer Eigenschaften ist notwendig, denn dies erlaubt es, die Möglichkeiten einer bestimmter Firewall-Implementierung abzuschätzen, sobald diese anhand des Modells klassifiziert ist.

Umsetzung der Designprinzipien. Wesentlich für die Eignung der gegebenen Firewall-Architekturklassen für die Unterstützung einer Multimedia-Applikation ist die Umsetzbarkeit der in Kapitel 3 beschriebenen Designprinzipien. Der Grad der Umsetzbarkeit dieser Prinzipien gibt Auskunft darüber, inwieweit eine Firewall-Architektur mit den für eine Multimedia-Applikation charakteristischen Eigenschaften umgehen kann.

Mit einem Proxy-System können diese Designprinzipien nicht umgesetzt werden. Eine Trennung von Signalisierungs- und Medienpfad ist bei dieser Architektur nicht möglich. Daher ist es nicht sinnvoll ein Proxy-System zu verwenden. Ein Hybridsystem kann prinzipiell verwendet werden, bestehende Systeme müssen dann aber in der Regel zunächst an die speziellen Anforderungen einer Multimedia-Applikation angepasst werden. Ein Hybridsystem besitzt ebenfalls Beschränkungen hinsichtlich der möglichen Gesamtleistung (siehe auch Kapitel 7). Betrachtet man nur die Designprinzipien, so ist es sinnvoll entweder ein verteilte Architektur oder eine endsystembasierte Architektur einzusetzen.

Wie gut die Designprinzipien umgesetzt werden können, kann jedoch nicht allein als Entscheidungsgrundlage für die Wahl einer geeigneten Architektur verwendet werden. Im Folgenden werden wesentliche Kriterien diskutiert, die bei einer Entscheidung für oder gegen eine Architektur ebenfalls zu beachten sind.

Realisierbarkeit. Architektur I (Proxy) ist in der Praxis am einfachsten umzusetzen. Es ist der Entwurf von nur einer einzelnen Firewall-Komponente notwendig. Diese kann in der Regel auf einem Standard-Betriebssystem aufsetzen. Architektur II (Hybridsystem) ist aufwendiger zu realisieren als Architektur I; es sind mehrere interagierende Komponenten zu entwerfen. Es kann kein Standard-Betriebssystem verwendet werden, da für die Realisierung der Filter/NAT-Komponente eine Modifikation des Betriebssystems nötig ist. Architektur III (verteilte Firewall) bietet den wesentlichen Vorteil, dass eine möglicherweise bereits vorhandene Komponente (z.B. ein Gatekeeper) für die Realisierung der *Application Control* verwendet werden kann. Dies reduziert den nötigen Implementierungsaufwand. Es ist aber zu beachten, dass die Erweiterung einer Infrastrukturkomponente oder die Ansteuerung eines Filters in der Praxis nicht immer umzusetzen ist. Architektur IV ist in der Praxis sehr schwer umzusetzen, da alle Endgeräte an die Firewall angepasst werden müssen.

Sicherheit. Komplexe Systeme sind in der Regel unsicherer als einfach aufgebaute Systeme [66]. Aus diesem Grund kann man prinzipiell annehmen, dass Architektur I die sicherste Variante darstellt. Bei Architektur II besteht eine klare Trennung zwischen Firewall und der Infrastruktur. Hier stellt die Aufteilung der Firewall auf verschiedene Komponenten eine zusätzliche Komplexität

dar, die möglicherweise zu zusätzlichen Sicherheitsproblemen führt. Bei Architektur III erhöht sich diese Komplexität des Firewall-Systems nochmals. Es ist ebenfalls zu beachten, dass durch die mögliche Einbindung von existierenden Infrastrukturkomponenten in die Firewall weitere Sicherheitslücken entstehen können. Bei Architektur IV sind Teile (*Application Control*) innerhalb des internen Netzwerkes angeordnet. Der Bereich der Firewall ist bei dieser Architekturen nicht ausschließlich auf die Netzgrenze beschränkt. Dies kann zu zusätzlichen Sicherheitsproblemen führen.

Betrieb. Für den Betrieb ist es von entscheidender Bedeutung, welchen Aufwand es mit sich bringt eine Firewall in ein Applikationsszenario einzubinden. Es ist in den meisten Fällen bei der Festlegung der Standards nicht vorgesehen worden, dass eine Firewall in den Kommunikationspfad eingebracht werden muss. Dieses Problem existiert insbesondere bei den ersten beiden Architekturen. Wird bei Architektur III (verteilte Firewall) eine Infrastrukturkomponente zur Realisierung der *Firewall Control* verwendet, so kann die Firewall sehr effizient in das Kommunikationsszenario eingebunden werden. Bei Architektur IV (endsystembasierte Firewall) kann die Firewall ebenfalls gut eingebunden werden, da die Applikation direkt mit der Firewall interagiert.

Ein weiterer wichtiger Aspekt ist, in welchem Zuständigkeitsbereich sich die einzelnen Firewall Komponenten befinden. Bei Architektur I und II werden alle Konfigurationen in einer zentralen Komponente durchgeführt. Dies gestaltet die Abstimmung der Konfiguration der *Application Control* mit der Konfiguration der *Firewall Control* relativ einfach. Bei den Architekturen III gestaltet sich dies insbesondere dann schwierig, wenn die *Application Control* innerhalb einer Infrastrukturkomponente untergebracht ist. In einem solchen Fall wird die Infrastrukturkomponente, die einem bestimmten Multimedia-Szenario zugeordnet ist, in der Regel von einer anderen Gruppe verwaltet als die Filter/NAT Komponente. Das gleiche Problem findet sich bei Architektur IV.

Beachtet man alle hier diskutierten Eigenschaften, und nicht nur die Umsetzbarkeit der Design Prinzipien, einer Firewall-Architektur ist es nicht möglich, die eindeutig beste Wahl einer Architektur anzugeben. Je nach festgelegtem Anforderungsprofil kann jede der beschriebenen Architekturen für einen bestimmten Anwendungsfall besonders geeignet sein.

Für die meisten Anwendungsfälle bietet sich aber die Verwendung einer verteilten Firewall (Architektur III) an. Das Designprinzip *Trennung von Signalisierungs- und Medienpfad* lässt sich dort besonders gut umsetzen. Gleichzeitig können diese Systeme noch mit relativ geringem Aufwand realisiert und sicher betrieben werden.

4.4 Existierende Firewall-Architekturen

In diesem Abschnitt werden verschiedene zur Zeit verwendete oder angedachte Firewall-Systeme bzw. Architekturen betrachtet. Diese Systeme und Architekturen lassen sich in die oben dargestellten Architekturklassen einordnen, wodurch eine generelle Aussage über deren Eignung im Multimedia-Umfeld gegeben werden kann. Die Betrachtung der existierender Arbeiten ermöglicht es ebenfalls vorhandene sinnvolle, bzw. noch fehlende Architekturelemente zu identifizieren.

4.4.1 Proxy

Viele der zur Zeit verwendeten Firewalls für Multimedia-Applikationen bestehen aus einem Proxy (Architektur I). Oft wird dieser Proxy zusammen mit einem statisch konfigurierten Paketfilter eingesetzt, d.h. es besteht keine Verbindung zwischen Filter und Proxy. Für den Entwurf eines Proxies für Multimedia-Applikationen gibt es keine festgelegten allgemeinen Vorschriften. Wie beschrieben, kann ein Proxy auch nicht effizient für die Unterstützung von Multimedia-Applikationen verwendet werden, da die notwendigen Designprinzipien nicht umgesetzt werden können. Dennoch wurden im Rahmen dieser Arbeit verschiedene Proxies untersucht (siehe [67] und [68]), um sinnvolle dort verwendeten Verfahren (z.B. das Einbinden der Firewall in ein Applikationsszenario) zu berücksichtigen. Im Folgenden werden nun zwei zur Zeit verwendete Systeme betrachtet.

Openh323Proxy [61]. Der OpenH323proxy kann als Firewall für das H.323-Protokoll verwendet werden und basiert auf dem OpenH323-Protokollstack [69] und wurde in [67] ausführlich untersucht und beschrieben. Der OpenH323Proxy stellt ein System dar, das im Wesentlichen auf einem H.323-Gatekeeper basiert, der zusätzlich (abweichend vom H.323-Standard) nicht nur den Signalisierungspfad verarbeitet, sondern auch den Medienpfad. Die Firewall wird dementsprechend, wie ein Gatekeeper, aktiv in ein Szenario eingebunden. Es besteht keine Möglichkeit besondere Sicherheitsüberprüfungen durchzuführen, wie sie in Kapitel 3 beschrieben sind.

RealNetworks Proxy [62]. RealNetworks hat eine Referenzimplementierung für eine Proxy-Firewall erstellt. Der Proxy wird wie ein im RTSP-Standard berücksichtigte Proxy-Cache Infrastrukturkomponente verwendet. Der Proxy bietet im Wesentlichen die selbe Funktionalität wie ein RTSP-Cache, allerdings ohne die Möglichkeit die übertragenen Daten wirklich zwischenzuspeichern. Es besteht auch hier keine Möglichkeit, besondere Sicherheitsüberprüfungen durchzuführen.

Zusammenfassung. Es hat sich gezeigt, dass beide beschriebenen Proxy-Systeme Probleme mit der Erfüllung der Dienstgüteanforderungen aufweisen (siehe auch Kapitel 7). Schon bei wenigen, parallel zu unterstützenden Sitzungen reicht die Leistungsfähigkeit des Medienpfades nicht mehr aus. Diese Beobachtung entspricht den in Abschnitt 4.2 getroffenen Aussagen bezüglich der Architekturklasse der Proxies. Die hier beschriebenen Proxy-Systeme basieren im Wesentlichen auf in den Szenarien verwendeten Infrastrukturkomponenten. Dies erklärt, warum diese Systeme

gut mit verschiedenen Applikationscharakteristika zusammenarbeiten und in verschiedene Szenarien gut eingebunden werden können.

Es empfiehlt sich demnach, ein Proxy-System nicht als Firewall für Multimedia-Applikationen einzusetzen. Es zeigt sich aber auch, dass beim Entwurf der die Signalisierung bearbeitenden Elemente einer Firewall die normalerweise in Infrastrukturkomponenten verwendeten Elemente zur Bearbeitung der Signalisierung als Vorlage dienen können.

4.4.2 Hybridsysteme

Kommerzielle Firewall-Systeme sind zur Zeit in den meisten Fällen als Hybridsysteme ausgeführt, die der Architektur II entsprechen. Wie bereits dargelegt, eignet sich eine solche Architektur nicht optimal, um Multimedia-Applikationen zu unterstützen. Es existieren auch für diese Architekturklasse wenig allgemeine Vorschläge, wie die Architektur eines Hybridsystems zu gestalten ist. Dies liegt daran, dass die durch die Industrie entwickelten und verwendeten Architekturen in der Regel nicht publiziert werden. In [70] wird beschrieben, wie ein Hybridsystem intern aufgebaut werden kann, um einzelne Datenpfade zu optimieren. Die dort dargestellten Verfahren und Optimierungen sind aber für Standard-Applikationen wie HTTP oder FTP beschrieben und lassen sich nicht ohne weiteres auf Multimedia-Applikationen übertragen. Im Rahmen dieser Arbeit wurden verschiedene Hybridsysteme untersucht (Siehe [67] und [68]). Im Folgenden werden die beiden zur Zeit am häufigsten eingesetzten Systeme beschrieben.

Cisco PIX [24]: Bei der von Cisco angebotenen Firewall Lösung PIX handelt es sich um eine spezialisierte Hardware mit entsprechender Software. Im Gegensatz zu vielen anderen Firewall-Produkten basiert die Firewall damit nicht auf Standard-Rechnerhardware und dem dazugehörigen Betriebssystem. Die Cisco PIX integriert einen Stateful-Filter, eine NAT-Komponente sowie Proxies für verschiedene Protokolle. Die Cisco PIX unterstützt verschiedene (Multimedia-) Applikationen und Protokolle, darunter auch das H.323-Protokoll. Die für dieses Protokoll durchgeführten Untersuchungen haben gezeigt, dass es extrem schwierig (bzw. unmöglich) ist ein PIX-System in ein bestehendes H.323-Szenario zu integrieren. Es ist ebenfalls zu beobachten, dass das PIX-System schlecht mit den in Kapitel 3 beschriebenen Applikationscharakteristika umgehen kann.

Checkpoint Firewall-1 [23]: Die Firewall-1 basiert auf einem Standard-Betriebssystem und besteht aus einem Stateful-Filter, einer NAT-Komponente sowie Proxies für verschiedene Protokolle. Bei durchgeführten Untersuchungen anhand des H.323-Protokolls wurden die selben Probleme wie bei der ebenfalls untersuchten PIX festgestellt. Es konnte in den durchgeführten Experimenten nur ein einziges Basisszenario (der direkte Ruf) unterstützt werden. Aus diesem Grund ist es nicht möglich, dieses System in verschiedenen H.323-Szenarien einzusetzen.

Zusammenfassung. Die bei den Hybridsystemen festgestellten Schwierigkeiten können auf die Art der Bearbeitung des Signalisierungspfades zurückgeführt werden. Die für die Bearbeitung des Signalisierungspfades verwendeten Proxy-Elemente müssen, um in die Firewall integriert werden zu können, an die dort vorhandenen Schnittstellen angepasst werden. Diese Schnittstellen sind

konzipiert, um Proxies für Standard-Applikationen (wie z.B. HTTP oder Telnet) aufzunehmen. Die verfügbaren Hybrid-Architekturen verfügen nicht über angepasste Schnittstellen, um einen für Multimedia-Applikationen angepassten Signalisierungs-Proxy aufzunehmen. Dies zeigt sich insbesondere bei dem Versuch, einen solchen Signalisierungs-Proxy in vorhandene Applikationsszenarien einzubinden.

Wie beschrieben können Hybridsysteme generell verwendet werden, um Multimedia-Applikationen zu unterstützen. Es ist aber eine Anpassung der Schnittstellen für das Einbringen von Signalisierungsproxies nötig.

4.4.3 Verteilte Systeme

Da die klassischen und zur Zeit eingesetzten Architekturklassen Proxy und Hybridsystem nicht für die Unterstützung von Multimedia-Applikationen eingesetzt werden können, gibt es Bestrebungen, neue Architekturen zu definieren. Die durchgeführten Arbeiten finden im Wesentlichen in den Standardisierungsgremien ITU und IETF statt. Diese Arbeiten wurden begonnen, als IP-Telefonie Szenarien (basierend auf H.323 oder SIP) in produktivem Umfeld eingesetzt werden sollten, und dauern zur Zeit noch an.

HFCI [71]. In der IETF wurde über die Festlegung eines *H.323 Firewall Control Interface* (HFCI) eine alternative Firewall-Architektur definiert. Kerngedanke dieses Vorschlages ist es, ein eine Filter-Komponente enthaltendes Gerät durch eine weitere, nicht direkt zur Firewall gehörende, Komponente anzusteuern. Damit entspricht die vorgeschlagene Architektur der Architekturklasse III (verteilte Firewall). Als steuernde Komponente war bei Entstehung dieses Vorschlages ein H.323-Gatekeeper geplant. Ziel des Vorschlag ist es, den Signalisierungs- und Medienpfad zu trennen, damit für die Implementierung des Signalisierungspfades vorhandene Infrastrukturkomponenten verwendet werden können.

Dieser Vorschlag wurde weiterentwickelt, da zum einen durch die resultierende Architektur keine NAT-Funktionalität bereitgestellt werden kann und zum anderen damit auch andere Multimedia-Protokolle die Architektur verwenden können. Diese Weiterentwicklung wird als *Firewall Control Protocol (FCP)*[1] bezeichnet und ist in [72] beschrieben. Dieses FCP definiert Nachrichten, die zwischen der Infrastrukturkomponente und der Filter/NAT-Komponente ausgetauscht werden können. Diese Nachrichten werden über UDP ausgetauscht. Zusätzlich wird beschrieben, wie eine Absicherung des verwendeten Kommunikationskanals erfolgt. Damit definiert dieser Vorschlag im Wesentlichen Teile der *Application Control* und der *Firewall Control* sowie den *Signalling Part* des in Abbildung 18 dargestellten Architekturmodells. Das hier beschriebene FCP wird zur Zeit von mehreren Filter/NAT-Komponenten der Firma ARAVOX [73] unterstützt.

[1] In der Literatur und auch in dieser Arbeit wird das für die Kommunikation zwischen Firewall-Komponenten verwendete Protokoll allgemein als Firewall Control Protocol (FCP) bezeichnet. Die hier beschriebene konkrete und standardisierte Ausprägung verwendet die gleiche Bezeichnung. Diese Überschneidung in der Namensgebung wurde bei der Standardisierung zwar angemerkt, aber letztendlich nicht berücksichtigt.

Midcom. Nach Festlegung des SIP-Standards und der zunehmenden Verbreitung von SIP basierten Geräten wurde auch in diesem Umfeld die Firewall-Problematik zunehmend betrachtet. Die verschiedenen Arbeiten im Umfeld von SIP und Firewalls werden innerhalb der IETF-Arbeitsgruppe *Midcom*[1] zusammengefasst. Ziel der Arbeitsgruppe ist es ein Framework zu entwickeln, das es Applikationen (oder Infrastrukturkomponenten) erlaubt, Policy-Änderungen an Netzwerkkomponenten zu übermitteln. Im Wesentlichen werden dabei Filter/NAT-Komponenten als Netzwerkkomponenten betrachtet. Durch die Trennung von Signalisierungs- und Medienpfad wird in *Midcom* angenommen, dass folgende Ziele erreicht werden können:

- Eine Erhöhung der Leistungsfähigkeit der Firewall
- Geringere Entwicklungskosten für Firewall-Software
- Geringere Betriebskosten
- Verbesserung der Unterstützung komplexer Protokolle durch Paketfilter
- Einfachere Entwicklung neuer Applikationen
- Konsolidierung von Managementfunktionen

Das aus diesen Überlegungen entwickelte Framework ist in [63] dargelegt. Dieses Dokument beschreibt im Wesentlichen den Aufbau der in Abbildung 18 dargestellten Elemente *Application Control*, *Firewall Control* und *Signalling Part*. Die daraus resultierende Firewall-Architektur entspricht damit der oben beschriebenen Architekturklasse III (verteilte Firewall), wobei die Definition aber auch Architektur IV ermöglicht.

In [63] ist nicht festgelegt, wie eine konkrete Umsetzung des zu verwendenden Protokolls für den *Signalling Part* vorgenommen werden muss. Die genaue Spezifikation dieses Protokolls wird zur Zeit in *Midcom* vorgenommen. Es existiert eine Beschreibung [74] der Anforderungen, die ein solches Protokoll erfüllen muss, außerdem wurden bereits verschiedene existierende Protokolle auf ihre Verwendbarkeit innerhalb des Frameworks [75] hin untersucht.

Die in *Midcom* durchgeführten Überlegungen basieren auf den für die Festlegung des HFCI durchgeführten Arbeiten (die jeweiligen Arbeiten haben zeitversetzt stattgefunden). Die in der IETF vorgeschlagenen Arbeiten sind deshalb als Weiterentwicklungen dieser Arbeiten zu sehen. Aktuelle Arbeiten auf diesem Gebiet finden in der IETF statt.

Zusammenfassung. Die aktuell in den Standardisierungsgremien angedachten Lösungen verfolgen das in Kapitel 3 festgelegte Designprinzip der Trennung von Signalisierungs und Medienpfad. Diese Arbeiten schlagen die Verwendung einer Infrastrukturkomponente zur Realisierung der *Application Control* vor. Dadurch kann das Firewall-System gut in bestehende Applikationsszenarien eingebracht werden. Dadurch, dass die Arbeiten zur Definition der hier vorgestellten Architekturen noch nicht abgeschlossen sind, bleiben noch zahlreiche Fragen offen.

Es ist noch nicht klar, welches Protokoll sich für den Informationsaustausch zwischen Applikationsseite und Firewall-Seite eignet. Ein weiteres Problem, das sich bei der Realisierung der vor-

[1] Midcom. Middlebox Communication Working Group. http://www.ietf.org/html.charters/midcom-charter.html.

geschlagenen Konzepte ergibt, ist die Tatsache, dass teilweise keine Infrastrukturkomponente gefunden werden kann, die sich für die Implementierung der *Application Control* eignet. In diesem Fall muss eine eigenständige Komponente entworfen werden, die in das Applikationsszenario eingebunden werden kann und sich zur Ansteuerung der Firewall-Seite eignet.

4.4.4 Endsystembasierte Firewall

Für endsystembasierte Firewalls existieren verschiedene Vorschläge. In der Praxis werden diese Systeme jedoch oft nicht verwendet. Alle Endysteme müssen entsprechend angepasst werden, dies ist sehr aufwendig und in der Praxis auch nicht immer möglich.

SOCKS [64]. SOCKS wurde entwickelt, um zu vermeiden, dass innerhalb einer Firewall für jedes Protokoll eine *Application Control* realisiert werden muss. Diese Funktionalität wird bei SOCKS in die einzelnen Applikationen verlegt. Damit dies nicht für jede Applikation einzeln geschehen muss, wird dies durch das Einführen einer zusätzlichen Abstraktionsschicht zwischen Applikations- und Netzwerkschicht erreicht. In der Praxis wird dies umgesetzt, indem verschiedene Netzwerkfunktionen (z.B. *bind(), connect()*), die in den Applikationen verwendet werden, durch entsprechende SOCKS-Funktionen ausgetauscht werden. Wird von einer Applikation ein Strom verwendet, was dem SOCKS-Framework über die Benutzung der veränderten Netzwerkfunktion angezeigt wird, so wird diese Kanalspezifikation an die Firewall-Seite übermittelt.

Das SOCKS-Framework ist ein etablierter Standard und wird auch in der Praxis eingesetzt. Das für eine Multimedia-Firewall festgelegte Designprinzip der Trennung von Signalisierungs und Medienpfad kann mit diesem Framework sehr gut umgesetzt werden. Es existieren aber folgende Probleme, weshalb sich SOCKS für Multimedia-Applikationen nicht eignet: Die zwischen Applikationsseite und Firewall-Seite ablaufende Kommunikation wird in-Band abgewickelt. Wird beispielsweise ein UDP-Kanal durch eine SOCKS-Applikation angekündigt wird zwischen *Application Control* und *Firewall Control* eine TCP-Verbindung aufgebaut. Über diese werden dann die notwendigen Informationen ausgetauscht. Danach können durch die Applikation die UDP-Daten gesendet werden, wobei jedes UDP-Paket einen zusätzlichen SOCKS-Header erhält. Dieser Header muss durch die Firewall-Komponenten entfernt werden, bevor das Paket weitergeschickt werden kann (der Empfänger des UDP-Paketes erwartet ein Standard-UDP-Paket ohne SOCKS Header). Durch diesen zusätzlichen Aufwand kann das SOCKS-Framework nicht effizient für Multimedia-Applikationen verwendet werden.

TARP [65]. Das als *Transient Addressing for Related Processes* (TARP) beschriebene Verfahren nutzt zur generischen Verlagerung der *Application Control* in die Endsysteme die Vorteile des IPv6 Protokolls. In IPv6 stehen deutlich mehr IP-Adressen zur Verfügung, als dies bei IPv4 der Fall ist. Geht man davon aus, dass genügend IPv6 Adressen zur Verfügung stehen, kann man festlegen, dass nicht jeder Host eine IP-Adresse verwendet, sondern jede Applikation auf einem Host eine eigene IP-Adresse verwendet. Das Problem, einzelne Kanalspezifikationen den Firewall-Komponenten mitteilen zu können kann umgangen werden, indem die Firewall Komponenten nur IP-Adressen betrachten, nicht aber einzelne Portnummern. Es muss den Firewall-Komponenten

nur mitgeteilt werden, für welche IP-Adressen eine Kommunikation erlaubt ist. Die Firewall kann erkennen, wenn der Kontrollkanal einer Multimedia-Applikation verwendet wird (Well Known Port). Ab diesem Zeitpunkt kann dann die Kommunikation der für die Signalisierung verwendeten IP-Adresse freigegeben werden. Damit können dann in Folge weitere dynamisch ausgehandelte Ströme zwischen den bei der Signalisierung beteiligten Systemen verwendet werden.

Das für eine Multimedia-Firewall festgelegte Designprinzip der Trennung von Signalisierungs- und Medienpfad kann mit diesem Framework umgesetzt werden. Neben der Problematik, dass dieses Verfahren nur für IPv6 verwendet werden kann, ergeben sich die folgenden Probleme: Es können nur dynamisch ausgehandelte Kanäle zwischen den an der Signalisierung beteiligten Systemen verwendet werden. Im Fall einer Multimedia-Applikation können die Signalisierungsströme aber von anderen Systemen durchlaufen werden, als die Medienströme. Darüber hinaus verletzt das beschriebene Framework das im Internet verwendete Schichtenmodell.

Zusammenfassung. Das in Kapitel 3 festgelegte Designprinzip lässt sich in dieser Architekturklasse gut umsetzen. Bei den hier betrachteten endsystembasierten Vorschlägen zeigt sich aber, wie schon bei den verteilten Systemen, dass die Festlegung der *Application Control*, *Firewall Control* und des *Signalling Parts* für die Verwendung von Multimedia-Applikationen in der vorliegenden Form nicht verwendet werden kann.

4.4.5 Nicht betrachtete Firewall-Architekturen

Es existieren bereits viele Arbeiten auf dem Gebiet Firewalls, der Fokus der meisten dieser Arbeiten liegt jedoch außerhalb der Fragestellung dieser Arbeit. Im Folgenden werden andere Arbeiten kurz vorgestellt, die sich ebenfalls mit der Definition, Festlegung oder Optimierung von Firewall-Architekturen beschäftigen, den Schwerpunkt aber auf andere Fragestellungen als die vorliegende Arbeit gelegt haben.

Tunnel. Viele Arbeiten beschäftigen sich damit, Verfahren zu entwickeln, die es ermöglichen eine Firewall zu tunneln. Im Wesentlichen wird dabei versucht, eine Infrastruktur so zu ändern oder zu betreiben, dass eine im Kommunikationsweg liegende Firewall für diese Infrastruktur nicht berücksichtigt werden muss. Dies kann beispielsweise erreicht werden, indem ein VPN zwischen zwei, durch eine Firewall getrennte, Teilen einer Infrastruktur verwendet wird. In der vorliegenden Arbeit wird eine Firewall als inhärenter Bestandteil eines Kommunikationsszenarios gesehen. Alle Ansätze, die eine Firewall tunneln, ziehen nach sich, dass die Firewall aus dem Szenario entfernt wird, weshalb solche Architekturansätze in dieser Arbeit nicht weiter betrachtet werden. Beispiele dieser Verfahren sind [76] und [77].

Dezentrale Firewall. Um die durch den Flaschenhals Firewall auftretenden Leistungsengpässe zu umgehen, werden Firewall-Architekturen vorgeschlagen, bei denen jeder Host über eine vollständige eigene Firewall verfügt. Nur eine global definierte Policy bleibt in diesem Ansatz als zentrale Komponente der Firewall bestehen. In dieser Arbeit wird zugrunde gelegt, dass eine

Firewall fest an den Übergangspunkt zwischen zwei Netzen gebunden ist; deshalb werden diese Vorschläge hier nicht näher betrachtet. Beispiel für dieses Verfahren ist [78].

Parallele Firewall. Viele Arbeiten beschäftigen sich damit, einzelne Firewalls parallel zu betreiben. Dies geschieht einerseits, um eine Leistungssteigerung zu erreichen [25], andererseits, um die Fehlertoleranz [79] zu erhöhen. Unter diesem Aspekt wird auch betrachtet, wie mehrere gleichartig betriebene Firewall-Systeme gemeinsam verwaltet werden können. In diesen Arbeiten geht es um den parallelen Betrieb ganzer Firewall Architekturen, nicht um die Optimierung einer einzelnen Architektur.

Andere Netze. Es gibt ebenfalls Vorschläge für Firewall Architekturen, die in anderen als IP basierten Netzen eingesetzt werden können. Die meisten Architekturvorschläge betreffen dabei die Realisierung von Firewall-Funktionalität für ATM-Netze (z.B. [80], [81]). Diese Arbeiten konzentrieren sich im Wesentlichen auf den Entwurf von Filterkomponenten. Zusätzlich lassen sich die dort gewonnenen Ergebnisse nicht vollständig auf IP basierte Firewalls übertragen.

Zukünftige Szenarien. Für die Zukunft wird eine starke Änderung des heute existierenden Internets angenommen. Um Firewalls an diese zukünftigen Anforderungen anzupassen, existieren bereits Vorschläge für angepasste Firewall-Architekturen.

Es wird beispielsweise angenommen, dass in Zukunft die Kommunikationssicherheit stärker berücksichtigt wird. Unter der Annahme, dass dafür IPsec verwendet wird, existieren Vorschläge für Firewall-Architekturen, die IPsec gesicherte Kommunikation effizient unterstützen (z.B. [24], [23]). Unter der Annahme, dass in Zukunft das Internet IPv6-basiert realisiert wird, existieren ebenfalls Vorschläge für IPv6 fähige Firewall Architekturen [65]. Ebenso existieren Architekturvorschläge für Firewalls, die Mobile IP [82] oder Multicast [83] unterstützen.

4.5 Zusammenfassung

In diesem Kapitel wurde ein Architekturmodell vorgestellt, das es ermöglicht Firewall-Architekturen hinsichtlich der für die Unterstützung von Multimedia-Applikation relevanten Architekturmerkmale zu ordnen. Es wurde anhand dieses Modells aufgezeigt, inwieweit sich verschiedene Architekturen für einen Betrieb im Multimediaumfeld eignen.

Es wurde gezeigt, dass sich eine verteilte Firewall, bzw. eine endsystembasierte Firewall, generell eignen. Betrachtet man die zur Zeit diskutierten und vorgeschlagenen Möglichkeiten, eine verteilte Firewall-Architektur oder eine endsystembasierte Firewall-Architektur umzusetzen, so ist festzustellen, dass eine passende Umsetzung des dafür notwendigen *Signalling Part* noch nicht existiert. Aus diesem Grund wird im folgenden Kapitel untersucht, wie dieses fehlende Element effizient realisiert werden kann. Es konnte ebenfalls festgestellt werden, dass der Aufbau der den Signalisierungspfad verarbeitenden Elemente der vorgeschlagenen Architekturen nicht optimal ist. In den nächsten Kapiteln wird deshalb ein Designvorschlag für diese Elemente vorgestellt, bei dem die bestehenden Probleme nicht auftreten.

Kapitel 5: Kommunikationsmechanismen verteilter Firewall-Architekturen

In diesem Kapitel wird betrachtet, wie das Element *Signalling Part* des in Kapitel 4 beschriebenen Architekturmodells effizient umgesetzt werden kann. Der Fokus liegt auf dem Entwurf dieses Elementes für die in Kapitel 4 beschriebenen Architekturklassen “Verteilte Firewall” und “Endsystembasierte Firewall”, da sich diese Klassen besonders für die Unterstützung von Multimedia-Applikationen eignen.

5.1 Motivation und Zielsetzung

Zur Umsetzung einer verteilten Firewall oder einer endsystembasierten Firewall muss der *Signalling Part* des in Kapitel 4 beschriebenen Architekturmodells so ausgelegt werden, dass die nötigen Informationen über ein Netzwerk transportiert werden können. Dazu ist es notwendig, ein geeignetes Protokoll für den *Signalling Part* festzulegen bzw. zu definieren. Zusätzlich muss festgelegt werden, wie der *Signalling Part* durch die *Application Control* und *Firewall Control* verwendet werden muss.

Betrachtet man den Entwurf des *Signaling Parts*, so stellt sich die Frage, ob für diesen Zweck ein neues Protokoll entworfen werden muss, oder auf bestehende Arbeiten zurückgegriffen werden kann. Diese Frage wird zur Zeit in *Midcom*[1] diskutiert. Es wurde untersucht, inwieweit sich verschiedene existierende Protokolle für diesen Zweck eignen. Es wurden dazu SNMP [84], RSIP [85], MEGACO [86], DIAMETER [87] und COPS [88] untersucht [75]. Ebenso wurden dort Vorschläge eingebracht, die neue Protokolle für diese Aufgabe festlegen (z.B. [89]). Weitere Vorschläge, die von anderen Arbeitsgruppen als *Midcom* erarbeitet wurden, sind in [71] und [72] beschrieben.

Aus diesen zur Zeit geführten Diskussionen wird deutlich, dass offenbar noch kein optimal geeignetes Protokoll identifiziert oder definiert werden konnte.

Motivation. Das Resource Reservation Protocol (RSVP) [90] wurde entwickelt, um die Belegung von Ressourcen zwischen Endsystemen und paketvermittelten Netzen auszuhandeln. Die durch eine Firewall bereitgestellten Schutzfunktionen können als eine durch das Netzwerk bzw. die innerhalb des Netzes vorhandene Firewall bereitgestellte Ressource verstanden werden. Wird eine endsystembasierte Firewall verwendet, so wird die Verwendung dieser Ressourcen zwischen End-

[1] Midcom. Middlebox Communication Working Group. http://www.ietf.org/html.charters/midcom-charter.html.

system und Netzwerk ausgehandelt. Es liegt daher nahe zu prüfen, ob RSVP sich als Protokoll für die Umsetzung des *Signalling Part* einer Firewall eignet, da die grundsätzlichen Aufgaben von RSVP dem eines Protokolls einer Firewall-Steuerung entsprechen.

Randbedingung. Für die folgenden Betrachtungen wird zunächst angenommen, dass nur im Internet gültige IP-Adressen verwendet werden, d.h., die Verwendung von NAT-Komponenten wird ausgeschlossen. In Abschnitt 5.6 wird aufgezeigt, inwieweit sich die gefundenen Ergebnisse auf Szenarien übertragen lassen, in denen auch NAT eingesetzt wird.

Zielsetzung. In diesem Kapitel wird gezeigt, dass es unter der oben beschriebenen Randbedingung möglich ist, RSVP zur Realisierung einer verteilten Firewall oder einer endsystembasierten Firewall zu verwenden. Es kann festgestellt werden, dass dies ohne wesentliche Veränderungen an RSVP möglich ist.

5.2 Funktionsweise des Resource Reservation Protocol (RSVP)

RSVP wurde zuerst in [91] beschrieben und ist mittlerweile innerhalb der IETF in RFC 2205 [90] festgeschrieben. RSVP wird verwendet, um die Belegung von Ressourcen zwischen Endsystemen und Komponenten paketvermittelter Netze auszuhandeln. Das RSVP-Modell sieht vor, dass ein Sender RSVP-fähige Router und Endsysteme über die Möglichkeit einer reservierungsbasierten Kommunikation informiert. Dazu verschickt der Sender eine sogenannte *PATH*-Nachricht.

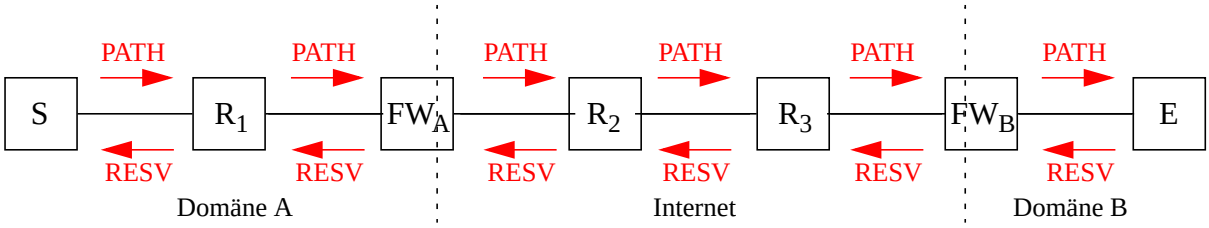


Abbildung 24: RSVP-Szenario

Die *PATH*-Nachricht trägt dabei die Verkehrsbeschreibung des Senders (*TSPEC*) und nimmt den selben Weg durch das Netzwerk, wie die Daten-Pakete der später ablaufenden Kommunikation. Der Empfänger antwortet auf den Erhalt der *PATH*-Nachricht mit einer *RESV*-Nachricht. Die *RESV*-Nachricht enthält dann eine Reservierungsbeschreibung (*RSPEC*). Diese Nachrichten werden durch die RSVP-fähigen Router verwendet, um eine Reservierung entlang des Pfades einzurichten.

RSVP Nachrichten. Nachfolgend ist der Aufbau der in RSVP vorhandenen Nachrichtenformate kurz beschrieben. Jede RSVP-Nachricht besteht grundsätzlich aus einem Header und einem Body, der aus einer variablen Anzahl an verschieden langen Elementen (Objekten) besteht. Die *PATH*-Nachrichten sind folgendermaßen aufgebaut:

```
<PATH MESSAGE> ::=          <COMMON HEADER>            [<INTEGRITY>]
                            <SESSION> <RESV_HOP>
                            <TIME_VALUE>
                            [<POLICY_DATA> ...]
                            [<SENDER_DESCRIPTOR>]
<SENDER_DESCRIPTOR> ::=     <SENDER_TEMPLATE>          <SENDER_TSPEC>
                            [<ADSPEC>]
```

Ist das Objekt *INTEGRITY* vorhanden, so muss es unmittelbar dem Header folgen. Die Reihenfolge der übrigen Objekte ist willkürlich, sollte sich aber an dieses Schema halten. Die Struktur der *RESV*-Nachrichten sieht folgendermaßen aus:

```
<RESV MESSAGE> ::=          <COMMON HEADER>            [<INTEGRITY>]
                            <SESSION> <RESV_HOP>
                            <TIME_VALUE>
                            [<RESV_CONFIRMATION>] [<SCOPE>]
                            [<POLICY_DATA> ...]
                            <STYLE> <FLOW DESCRIPTOR LIST>
<FLOW DESCRIPTOR LIST> ::= <EMPTY> |
                            <FLOW DESCRIPTOR LIST> <FLOW DESCRIPTOR>
```

Die Formate der übrigen Nachrichten-Typen (*PATHTEAR*, *RESVTEAR*, *PATHERR*, *RESVERR*, etc.) sind ähnlich aufgebaut. Detaillierte Angaben hierzu befinden sich in [90]. Alle RSVP-Nachrichten verwenden ein eigenes Protokoll (Protokollnummer 46); kann dies nicht unterstützt werden können auch RSVP-Nachrichten in UDP gekapselt transportiert werden.

Beispiel. Folgendes Beispiel soll die Funktionsweise von RSVP verdeutlichen. Zunächst wird angenommen, dass alle in Abbildung 24 dargestellten Netzwerkelemente gewöhnliche RSVP-fähige Router darstellen. In späteren Beispielen wird angenommen, dass FW_A und FW_B RSVP-fähige Firewall-Systeme sind. Es wird angenommen, dass der Sender einen UDP-Strom (Sender Port = P_S, Sender IP-Adresse = IP_S) an den Empfänger (Empfänger Port = P_E, Empfänger IP-Adresse = IP_E) senden möchte und eine Reservierung für diesen Strom durchgeführt werden soll. Folgende Mechanismen werden verwendet:

- **PATH:** S sendet eine *PATH*-Nachricht an E. Diese Nachricht wird entsprechend dem gesetzten IP-Routing durch das dazwischen liegende Netz transportiert. Die RSVP-fähigen Router werten vor dem Weiterleiten der *PATH*-Nachricht diese aus und legen für diesen Pfad einen *path state* an. Das *SESSION*-Objekt beschreibt das Ziel des Stroms (UDP, P_E, IP_E), das *HOP*-Objekt kennzeichnet den RSVP-Router, der die Nachricht übermittelt hat. Das *SENDER_TEMPLATE* und *SENDER_TSPEC* enthalten die Quelle des Stroms (P_S,IP_S) und die Verkehrsbeschreibung des Stroms. *SESSION*, *HOP* und *SENDER_TEMPLATE* identifizieren eindeutig den *path state*. Das *TIME_VALUES*-Objekt beschreibt den Zeitpunkt, an dem der *path state* seine Gültigkeit verliert (Timeout). Nachdem die Nachricht durch einen Router bearbeitet wurde wird sie nach Aktualisierung des *HOP*-Objektes an den nächsten Router weitergeleitet. Ist ein Router nicht RSVP-fähig wird die Nachricht wie ein normales IP-Paket weitergeleitet.
- **RESV:** Wird die *PATH*-Nachricht vom Empfänger empfangen, kann dieser sich entscheiden eine Reservierung durchzuführen. Dazu sendet der Empfänger eine *RESV*-Nachricht an den durch das *HOP*-Objekt der empfangenen *PATH*-Nachricht identifizierten Router. Diese enthält wiederum das *SESSION*-Objekt, wodurch der entsprechende *path state* identifiziert werden kann. Existiert ein *path state* für die Sitzung, kann die Reservierung durchgeführt werden und es wird ein *reservation state* angelegt. In der Nachricht ist ebenfalls ein *TIME_VALUES*-Objekt zum Setzen eines Timeouts für die Reservierung angegeben sowie weitere Felder, die die Art der Reservierung spezifizieren. Nachdem die Nachricht durch einen Router bearbeitet wurde, wird sie jeweils an den nächsten Router weitergeleitet bis sie zum Sender S gelangt.

Die einzelnen RSVP-fähigen Router sind nun so konfiguriert, dass der UDP-Strom zwischen Sender und Empfänger die nötigen Ressourcen erhält, damit die für den Strom notwendigen Bedingungen eingehalten werden können. Der Sender kann nun mit dem Senden der Daten beginnen.

5.3 Anforderungen an ein Firewall Control Protocol (FCP)

Es existieren verschiedene Arbeiten, die beschreiben, welchen Anforderungen ein Protokoll für die Realisierung des *Signalling Part* erfüllen sollte ([74], [92]). Die dort beschriebenen Anforderungen können in zwei wesentliche Teilbereiche untergliedert werden: Sicherheitsanforderungen und Protokollanforderungen. In den folgenden Abschnitten werden diese Anforderungen verwendet, um zu prüfen, ob sich RSVP grundsätzlich als Firewall Control Protocol verwendet werden kann.

Sicherheitsanforderungen. Da das Protokoll verwendet wird, um eine Firewall zu steuern, muss das Protokoll selbst ebenfalls bestimmte Schutzmechanismen zur Verfügung stellen. Es werden folgende Sicherheitsmechanismen gefordert:

1. Es muss eine sichere Kommunikation über unsichere Netze möglich sein.
2. Eine beiderseitige Authentifizierung muss erfolgen. Die *Application Control* muss sich bei der *Firewall Control* authentifizieren und umgekehrt.
3. Die transportierten Nachrichten müssen gegen Veränderung gesichert werden. Ein Integritätsschutz ist notwendig.
4. Die transportierten Nachrichten müssen vor dem Abhören geschützt werden. Ein Schutz der Vertraulichkeit ist notwendig.
5. Schutz vor Replay Angriffen muss gewährleistet sein.

Protokollanforderungen. Neben den generellen Anforderungen an ein Signalisierungsprotokoll (z.B. deternimistisches Protokollverhalten, Handhabung unbekannter Nachrichtentypen) müssen folgende wesentlichen Eigenschaften erfüllt werden:

1. Eine Instanz einer *Application Control* sollte mit mehreren *Firewall Control* Instanzen interagieren können. Es ist denkbar, dass die Kommunikation mit verschiedenen Zielen über verschiedene Firewall-Systeme geführt wird. Ebenfalls kann es möglich sein, dass mehrere Firewall-Systeme in einem Kommunikationsweg liegen. Ebenso sollte eine Instanz einer *Firewall Control* mit mehreren Instanzen der *Application Control* interagieren können, da normalerweise mehrere Applikationen gleichzeitig über ein Firewall-System kommunizieren.
2. Um flexibel neuen Anforderungen nachkommen zu können, ist es notwendig, dass die Protokollsemantik eines verwendeten Protokolls einfach erweitert werden kann.
3. Es wird ein Soft-State bzw. Refresh-Mechanismus benötigt. Die an eine Firewall-Komponente übermittelten Informationen (z.B. Filterspezifikationen) dürfen dort nur eine gewisse Zeit Gültigkeit besitzen. Werden die Informationen durch die *Application Control* nicht periodisch aktualisiert, so verlieren sie ihre Gültigkeit. Alternativ müssen Mechanismen vorhanden sein, die es der *Application Control* oder der *Firewall Control* ermöglichen, (asynchron) die Gültigkeit einer Filterspezifikation wieder explizit aufzuheben.
4. Die wesentlichen Informationen, die zur Steuerung einer Firewall verwendet werden, sind Filterspezifikationen in Form eines 5-Tupels. Dementsprechend muss das verwendete Protokoll diese Informationen transportieren können.

5. Wurde durch die *Application Control* eine Information an die *Firewall Control* übermittelt, so muss eine entsprechende positive oder negative Bestätigung an die *Application Control* übermittelt werden.
6. Das Protokoll muss eine entsprechende Fehlersicherheit besitzen. Mit Fehlfunktionen einer Instanz der *Application Control* bzw. *Firewall Control* muss umgegangen werden können. Es muss beispielsweise möglich sein, dass gebundene Ressourcen nach Versagen eines Elementes wieder freigegeben werden können. Auftretende Fehler müssen ebenfalls entsprechend durch Fehler-Nachrichten angezeigt werden.
7. Das Protokoll soll es ermöglichen, mehrere Informationen (z.B. mehrere Kanalspezifikationen) in einer Nachricht zu verschicken.

Neben den hier aufgezählten Anforderungen existieren weitere Anforderungen, die eine Steuerung von NAT-Komponenten betreffen. Entsprechend der in Abschnitt 5.1 beschriebenen Annahme, dass nur global gültige IP-Adressen verwendet werden, werden diese Anforderungen hier nicht berücksichtigt.

5.4 Untersuchung der Verwendbarkeit von RSVP als FCP

Damit RSVP als FCP verwendet werden kann, muss RSVP die im vorigen Abschnitt beschriebenen Sicherheits- und Protokollanforderungen erfüllen. Ob dies der Fall ist, wird im Folgenden analysiert.

5.4.1 Umsetzung der Sicherheitsanforderungen

Soll RSVP als Signalisierungsprotokoll für Firewall-Systeme Verwendung finden, muss gewährleistet sein, dass das Protokoll selbst entsprechende Sicherheitskriterien erfüllt. Dabei können folgende Aspekte unterschieden werden: Authentifizierung, Integrität und Vertraulichkeit. Im Folgenden wird untersucht, ob - und wenn ja wie - diese Aspekte innerhalb von RSVP umgesetzt werden können.

Authentifizierung. Bevor eine empfangene RSVP-Nachricht verarbeitet werden kann, muss überprüft werden, ob diese auch wirklich von dem erwarteten Sender abgeschickt wurde. Für das Hop-by-Hop-basierte RSVP bedeutet dies, dass ein Knoten sich beim jeweilig nächsten Knoten authentifizieren muss. Innerhalb von RSVP wird diese Hop-by-Hop Authentifizierung zusammen mit dem zur Integritätsprüfung verwendeten Mechanismus gelöst und wird deshalb im nächsten Abschnitt beschrieben. Durch den Hop-by-Hop Charakter der Authentifizierung kann ein RSVP-Knoten jeweils nur die Identität des Senders der RSVP-Nachricht überprüfen, nicht aber die Identität des Initiators der Nachricht. Soll ebenfalls die Identität des Initiators einer RSVP-Nachricht überprüft werden, kann das im Folgenden beschriebene Verfahren verwendet werden.

In [93] wird ein *POLICY*-Objekt für RSVP definiert, um mit RSVP-Zugangs und Benutzungsrechte (entsprechend einer Policy) umzusetzen zu können. In [94] wird darauf aufbauend beschrieben, wie *POLICY*-Objekte zur Identifizierung von Benutzern und Applikationen verwendet werden können. Die *POLICY*-Objekte können in die verwendeten RSVP-Nachrichten eingebettet werden.

Das Grundformat eines *POLICY*-Objektes ist in Abbildung 25 dargestellt. In der *Option List* können bekannte RSVP-Objekte verwendet werden, allerdings haben diese im Kontext des *POLICY*-Objektes eine andere Bedeutung. So kennzeichnet beispielsweise das *HOP*-Objekt den nächsten Router, der policy fähig ist, nicht den nächsten RSVP-fähigen Router. Die *POLICY_ELEMENTS* enthalten die zur Umsetzung einer Policy notwendigen Informationen. Das interne Format der *POLICY_ELEMENTS* muss nur durch die Komponenten interpretiert werden, die für die Umsetzung der darin enthaltenen Informationen sorgen.

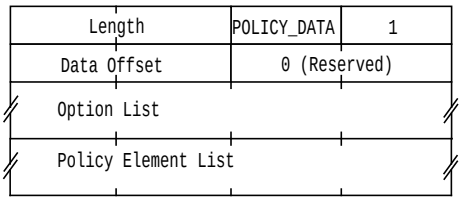


Abbildung 25: RSVP POLICY_DATA

In [94] wird ein *POLICY_ELEMENT AUTH_DATA* (siehe Abbildung 26) definiert, mit dessen Hilfe die Repräsentation von Identitäten in RSVP durchgeführt werden kann. Über das Feld *P-Type* wird der Typ des Authentifizierungsschemas festgelegt. Zur Zeit sind zwei Schemata festgelegt: *AUTH_USER* zur Identifikation von Benutzern und *AUTH_APP* zur Identifikation von Applikationen. Jedes *AUTH_DATA*-Objekt enthält eine Reihe von sogenannten *AUTHENTICATION_ ATTRIBUTES*. Innerhalb der Liste der Attribute können bisher 4 vordefinierte *AUTHENTICATION_ ATTRIBUTES* Platz haben:

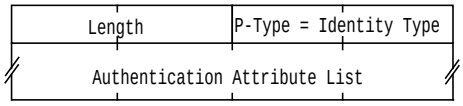


Abbildung 26: RSVP AUTH_DATA

- **Policy Locator:** Dieses Attribut wird zur eindeutigen Unterscheidung von Benutzern und Anwendungen mittels eines *Distinguished Name* verwendet.
- **Credential:** Dieses Attribut dient als Nachweis der Identität. Dieses Attribut kann ein Kerberos-Ticket oder ein digitales Zertifikat sein.
- **Digital Signature:** Die Signatur muss als letztes Feld eingefügt werden und signiert das gesamte Paket.
- **Policy Error Message:** Dieses Objekt wird in *PATHERR* oder *RESVERR*-Nachrichten eingefügt.

Ein Empfänger einer RSVP-Nachricht kann so den ursprünglichen Absender mit Hilfe eines *POLICY*-Objektes identifizieren.

Integrität. Zur Sicherstellung der Integrität der RSVP-Nachrichten und zur Authentifizierung des Absenders der Nachricht kann in RSVP das in [95] beschriebene Verfahren verwendet werden. Dazu wird das in Abbildung 27 dargestellte *INTEGRITY*-Objekt definiert. Der Schutzmechanismus beruht auf einem authentizierten Extrakt der Nachricht, einem Message Digest, das aus einem geheimen Schlüssel und dem Hashwert der Nachricht errechnet wird. Als kryptographische Hashfunktion muss mindestens HMAC-MD5 von jedem RSVP-Knoten unterstützt werden, die Verwendung anderer Hashfunktionen ist aber ebenso möglich. Innerhalb des *INTEGRITY*-Objektes wird das Feld *Key Identifier* verwendet, um eine *Security Asssociation* festzulegen. Ein Empfänger einer Nachricht kann über die Absendeadresse der Nachricht und den dort eingefügten 48-bit langen *Key Identifier* den passenden Algorithmus und Schlüssel zur Überprüfung der erhaltenen Nachricht bestimmen. Um Schutz vor Replay-Attacken bieten zu können, wird eine Sequenznummer verwendet. Der Sender erzeugt eine monoton steigende Zahlenfolge und der Empfänger akzeptiert nur dann eine Nachricht, wenn die Folgenummer dieses Pakets größer ist als die der vorherigen Nachricht. Eine korrekte Prüfung des in der Nachricht enthaltenen *Keyed Message Digest* ist durch einen Empfänger nur möglich, wenn die Nachricht auf dem Transportweg nicht modifiziert wurde und der Absen-

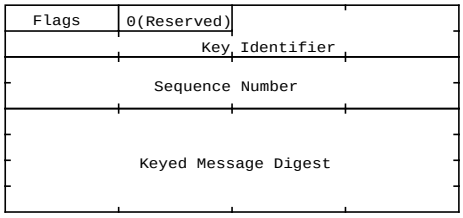


Abbildung 27: RSVP INTEGRITY

der der Nachricht über den korrekten Schlüssel verfügt. Damit kann eine Authentifizierung des Absenders sowie ein Integritätsschutz für die RSVP-Nachrichten umgesetzt werden.

Vertraulichkeit. Es ist möglich, die Vertraulichkeit von RSVP-Nachrichten über IPsec [96] zu gewährleisten. Dazu ist es aber notwendig, wie in [97] beschrieben wird, verschiedene neue Objekte innerhalb von RSVP zu definieren. Es muss beispielsweise ein neues *SESSION*-Objekt eingeführt werden. In diesem werden die TCP- oder UDP-Ports durch den IPsec Security Parameter Index (SPI) ersetzt. Dafür wird ein virtueller Port *vDstPort* (virtual Destination Port) eingeführt. Ein RSVP-Sitzung besteht dann aus dem 3-Tupel (*DestAddress, protocol ID, vDstPort*). Zusätzlich kann auch über die Verwendung von IPsec die Authentifizierung und der Integritätsschutz realisiert werden. Dementsprechend kann dieser Ansatz als alternative Möglichkeit zu den zuvor beschriebenen Möglichkeiten gesehen werden.

Es ist zu beachten, dass viele Gründe dafür sprechen, dass bei einem Transport von RSVP-Nachrichten auf die Umsetzung des Aspekts Vertraulichkeit verzichtet werden kann. Die Nachrichten werden verwendet, um den Transportweg für die nachfolgend gesendeten Daten vorzubereiten. Durch Abhören der RSVP-Nachrichten kann ein Angreifer im Wesentlichen ermitteln, welche Ströme in Zukunft verwendet werden. Diese Information erhält der Angreifer aber in jedem Fall, wenn auch zeitversetzt. Der Angreifer kann dazu den ablaufenden Kommunikationsverkehr beobachten. Dies ist ihm möglich, wenn er in der Lage ist, die RSVP-Nachrichten einzusehen, da RSVP-Nachrichten und Daten den gleichen Weg durch das Netz nehmen.

Beispiel für eine abgesicherte RSVP-Kommunikation. Verzichtet man auf die Vertraulichkeit, so bietet es sich an, die oben beschriebene Hop-by-Hop Methode zur Umsetzung von Authentifizierung und Integrität zu verwenden. Dazu müssen - in den abzusichernden Bereichen - an die an der RSVP-Kommunikation teilnehmenden Systeme entsprechende Schlüssel verteilt werden. Dies kann, wie in [95] beschrieben, entweder manuell oder über passende Protokolle geschehen. In dem in Abschnitt 5.2 dargestellten Referenzszenario kann eine Authentifizierung und Integritätsschutz dann folgendermaßen umgesetzt werden:

- Die in der Domäne A verwendeten Systeme S (Sender), R_1 (interner RSVP-fähiger Router) und FW_A (Firewall) werden so konfiguriert, dass im internen Netz nur gesichert RSVP-Nachrichten ausgetauscht werden. In alle auszutauschenden Nachrichten wird das *Integrity*-Objekt eingefügt. Eine durch S gesendete RSVP-Nachricht wird durch R_1 geprüft, bei erfolgreicher Prüfung wird die Nachricht an FW_A weitergeschickt. Bei erfolgreicher Prüfung dieser Nachricht durch FW_A kann diese dann zur Steuerung der Firewall FW_A verwendet werden. Wird diese Nachricht dann an R_2 weitergeleitet, kann auf die Umsetzung der Sicherheitsmechanismen verzichtet werden. Wie später beschrieben, werden zur Steuerung der Firewall nur gesicherte Nachrichten, die aus dem internen Netz kommen, akzeptiert.

Ist diese eben beschriebene Hop-by-Hop Authentifizierung nicht ausreichend, so kann zusätzlich das *POLICY*-Objekt verwendet werden, um S gegenüber FW_A auszuweisen. Diese Authentifizierung wird dann folgendermaßen durchgeführt:

- In eine durch S zu sendende Nachricht wird ein *POLICY*-Objekt eingefügt. Beispielsweise kann die verwendete Applikation den Benutzer veranlassen, sich durch Eingabe Passwortes gegenüber einem Kerberos-Server zu authentifizieren. Die so ermittelten *Credentials* werden in dem *POLICY*-Objekt zusammen mit dem Benutzernamen abgelegt. Wird diese Nachricht nun durch FW_A empfangen, kann die Firewall mit Hilfe des Kerberos-Servers überprüfen, ob die empfangene Nachricht durch den angegebenen Benutzer verschickt wurde.

Zusammenfassung. Wie gezeigt wurde, existieren für alle Aspekte der Absicherung eines Kommunikationskanals auch in RSVP entsprechende Mechanismen und Erweiterungsmöglichkeiten. Betrachtet man nun, inwieweit die beschriebenen Sicherheitsanforderungen umgesetzt werden können, so ist festzustellen, dass alle Sicherheitsanforderungen umgesetzt werden können.

Es ist allerdings zu beachten, dass Punkt 1 nur dann erfüllt werden kann, wenn das zu überbrükkende unsichere Netz keine RSVP-Nachrichten verarbeitet. Ist dem in Abbildung 24 zwischen S und FW_A gelegenen Router R_1 nicht zu trauen, kann, wenn R_1 ein RSVP-fähiger Router ist, kein Integritätsschutz gewährleistet werden. R_1 muss zur Verarbeitung der RSVP-Nachricht diese verändern.

5.4.2 Umsetzung der Protokollanforderungen

Im Folgenden wird untersucht, ob sich die Protokollanforderungen generell mit RSVP umsetzen lassen. Prinzipiell handelt es sich bei RSVP um ein seit längerem festgeschriebenes und erprobtes Signalisierungsprotokoll, dementsprechend werden allgemeine Anforderungen (z.B. deterministisches Verhalten) bereits durch RSVP erfüllt.

Protokollanforderungen. Für die speziellen Anforderungen gilt:

1. Eine Instanz einer *Application Control* kann mit mehreren *Firewall Control* Instanzen kommunizieren. Dies gilt einerseits, wenn verschiedene Ziele über verschiedene Firewall-Systeme erreicht werden. Die RSVP-Nachrichten nehmen dann entsprechend des verwendeten IP-Routings den selben Weg, wie die dann folgenden Daten. Dementsprechend wird jeweils mit der passenden Instanz der *Firewall Control* kommuniziert. Andererseits kann mit mehreren Instanzen einer *Firewall Control* auf einem Kommunikationspfad interagiert werden, da die RSVP-Nachrichten durch jedes Firewall-System auf dem Kommunikationspfad verarbeitet werden können.
   Ebenfalls können mehrere RSVP-fähige Endsysteme (*Application Control*) gleichzeitig mit RSVP-fähigen Routern oder Firewall Systemen (*Firewall Control*) kommunizieren.
2. RSVP-Nachrichten sind im Aufbau flexibel. Es ist jederzeit möglich eigene Objekte innerhalb existierender Nachrichten zu definieren, oder auch komplett neue Nachrichtentypen festzulegen.
3. RSVP verwendet einen Soft-State Mechanismus. Werden vorgenommene Reservierungen nicht aktualisiert, so werden die damit verbundenen Zustände durch die RSVP-fähigen Netzkomponenten aufgehoben.
4. Ein Strom wird innerhalb von RSVP im Wesentlichen durch ein 5-Tupel beschrieben.

5. Es existieren in RSVP Mechanismen, um negative Bestätigungen für eine vorgenommene Reservierung durchzuführen. Dies erfolgt mit Hilfe von *PATHERR*- oder *RESVERR*-Nachrichten. Eine positive Bestätigung einer Reservierung erfolgt durch ein *RESVCONF*-Objekt innerhalb einer *RESV*-Nachricht.
6. Fehler können durch entsprechende *PATHERR*- oder *RESVERR*-Nachrichten angezeigt werden. Wesentliches Element der Fehlerbehandlung ist der Soft-State. Werden bestimmte Ressourcen durch auftretende Fehler nicht mehr verwendet, so werden diese durch Timeouts freigegeben.
7. Es ist möglich, mehrere RSVP-Nachrichten innerhalb einer RSVP Bundle Nachricht zusammenzufassen. Diese RSVP-Erweiterung ist in [98] beschrieben.

Weitere Vorteile. Neben der Möglichkeit, die festgeschriebenen Anforderungen mit RSVP zu erfüllen, beinhaltet die Verwendung von RSVP als Firewall Control Protocol weitere Vorteile.

Ein wesentliches Problem bei der Verwendung einer endsystembasierten Firewall besteht darin, dass die Endsysteme wissen müssen, wo die zu steuernden Firewall-Komponenten auf dem Kommunikationsweg liegen. Es wurden deshalb verschiedene Arbeiten durchgeführt, die sich mit der Detektion von Firewall-Komponenten beschäftigen (siehe z.B. [99] und [100]), damit diese dann nachfolgend durch das Firewall Control Protocol angesteuert werden können. Verwendet man RSVP als Firewall Control Protocol, so kann auf diese Detektion verzichtet werden. Die RSVP-Nachrichten folgen dem Kommunikationsweg und werden durch Firewall-Komponenten, die sich auf diesem Weg befinden, interpretiert.

Ein weiterer Vorteil von RSVP ist, dass sowohl IPv4 als auch IPv6 unterstützt wird. Damit sind keine Anpassungen des Protokolls nötig, wenn IPv6 Netze unterstützt werden sollen.

Zusammenfassung. Es wurde gezeigt, dass sich die beschriebenen Protokollanforderungen mit Hilfe von RSVP umsetzen lassen. Damit ist es prinzipiell möglich, RSVP als Firewall Control Protocol zu verwenden. In [101] wurde bereits gezeigt, dass eine RSVP-Signalisierung auch für eine sehr große Anzahl paralleler Ströme effizient verwendet werden kann. Somit kann aus diesen bereits vorhandenen Ergebnissen geschlossen werden, dass ein RSVP-FCP auch hohen Leistungsanforderungen genügen wird.

5.5 Entwurf und Implementierung eines RSVP-FCP

Nachdem gezeigt wurde, dass RSVP sich prinzipiell zur Realisierung einer Firewall-Steuerung eignet, werden im Folgenden Entwurf und Implementierung eines RSVP-FCP beschrieben.

5.5.1 Entwurf eines RSVP-FCP

Als Grundlage für den Entwurf eines RSVP-FCP dient das in Abbildung 24 beschriebene Kommunikationsszenario. Es wird angenommen, dass der Sender S einen UDP-Strom (Sender Port = P_S, Sender IP-Adresse = IP_S) an den Empfänger (Empfänger Port = P_E, Empfänger IP-Adresse = IP_E) senden möchte. Dazu muss eine entsprechende Konfigurationsanpassung der Firewall FW_A und FW_B durchgeführt werden, damit die gewünschte Kommunikation erfolgen kann.

Basisfunktionalität. Folgende Mechanismen werden verwendet, um die Firewall-Systeme an die gewünschten Kommunikationsparameter anzupassen:

- Der Sender erzeugt eine *PATH*-Nachricht, um den verwendeten UDP-Strom anzukündigen. Diese Nachricht wird an den Empfänger E adressiert. Die erzeugte RSVP-Nachricht entspricht einer Standard-*PATH*-Nachricht und enthält unter anderem ein *SESSION*-Objekt zur Beschreibung des Zieles des Stroms (UDP, P_E, IP_E) sowie das *SENDER_TEMPLATE* zur Beschreibung der Quelle des Stroms (P_S,IP_S).
- Die *PATH*-Nachricht wird an den nächsten Router weitergeleitet (R_1). Ist R_1 RSVP-fähig, kann die *PATH*-Nachricht innerhalb des Routers wie in einem gewöhnlichen RSVP Szenario verwendet werden. Nach Bearbeitung wird die *PATH*-Nachricht an den nächsten Router, in diesem Fall FW_A , weitergeleitet.
- FW_A verwendet nun die Nachricht, um den *path state* zu etablieren. Danach wird die Nachricht an den Router R_2 weitergeleitet.
- Nun wird die *PATH*-Nachricht entlang der Router R_2 und R_3 an FW_B weitergeleitet. Sind R_2 und R_3 RSVP-fähige Router, so kann die *PATH*-Nachricht von diesen Knoten im herkömmlichen RSVP-Sinn dort verwendet werden.
- FW_B empfängt die *PATH*-Nachricht und etabliert den *path state*. Danach wird die Nachricht an den Empfänger E weitergeleitet.
- Der Empfänger antwortet nun mit einer Standard-*RESV*-Nachricht. Diese enthält unter anderem das *SESSION*-Objekt. Diese Nachricht wird nun zuerst an FW_B gesendet.
- FW_B kann die *RESV*-Nachricht durch das *SESSION*-Objekt dem passenden *path state* zuordnen. Unter Verwendung des *SESSION*-Objektes und des *Sender Template* wird ein 5-Tupel gebildet. Dieses wird verwendet, um eine entsprechende Konfigurationsänderung innerhalb der Firewall zu etablieren. Diese Konfigurationsänderung bleibt innerhalb der Firewall solange aktiv, bis entweder durch E signalisiert wird, dass diese nicht mehr benötigt wird oder die Gültigkeit dieser Regel durch den Timeout erlischt (Soft-State). Die *RESV*-Nachricht wird an R_3 weitergeleitet.

- R_3 und R_2 können die *RESV*-Nachricht im herkömmlichen RSVP-Sinn verwenden. R_2 leitet die Nachricht an FW_A weiter.
- FW_A kann die *RESV*-Nachricht durch das *SESSION*-Objekt dem passenden *path state* zuordnen. Für den angekündigten UDP-Strom kann nun die Konfigurationsänderung innerhalb der Firewall durchgeführt werden. Diese bleibt aktiv, bis entweder durch S signalisiert wird, dass diese nicht mehr benötigt wird, oder die Gültigkeit dieser Regel durch den Timeout erlischt (Soft-State). Die *RESV*-Nachricht wird an R_1 weitergeleitet.
- R_1 kann die *RESV*-Nachricht im herkömmlichen RSVP-Sinn verwenden. R_1 leitet die Nachricht an S weiter.
- Die Firewall FW_A und FW_B sind nun so konfiguriert, dass der gewünschte UDP-Strom zwischen S und E verwendet werden kann.

Sicherheitsbetrachtung. Für die oben beschriebene Kommunikation müssen ebenfalls Überlegungen hinsichtlich der Sicherheit angestellt werden. Zunächst einmal kann nicht davon ausgegangen werden, dass die in Abbildung 24 beschriebenen Domänen A und B einer gemeinsamen Verwaltungsstruktur unterliegen. Demnach kann auch nicht angenommen werden, dass eine Authentifizierung eines Benutzers oder einer Komponente aus der Domäne A gegenüber einem Benutzer oder einer Komponente innerhalb der Domäne B oder umgekehrt erfolgen kann. Weiterhin muss angenommen werden, dass die jeweilige Domäne nur die eigenen Netzwerkkomponenten unter Kontrolle hat und auch nur deshalb diesen Vertrauen schenken wird. Jede Domäne wird dementsprechend ein eigenes Sicherheitsmodell für die RSVP-Kommunikation verwenden. Für die Verarbeitung der Nachrichten im oben beschriebenen Szenario ergibt sich damit folgendes:

- Nachrichten, die von FW_A und FW_B auf dem jeweiligen internen Interface entgegengenommen werden oder über dieses Interface gesendet werden, müssen ein *INTEGRITY*-Objekt zur Sicherstellung der Integrität und der Authentifizierung verwenden.
- Nachrichten, die von FW_A und FW_B auf dem jeweiligen externen Interface entgegengenommen werden, oder über dieses Interface gesendet werden, bedürfen keiner Sicherung.
- Ist der Router R_1 innerhalb der Domäne A RSVP-fähig, so müssen alle von ihm angenommenen und gesendeten RSVP-Nachrichten über ein *INTEGRITY*-Objekt verfügen.
- Alle RSVP-Nachrichten, die durch die Endsysteme (Empfänger und Sender) angenommen oder gesendet werden, müssen über ein *INTEGRITY*-Objekt gesichert werden.
- Alle RSVP-Nachrichten, die durch die Endsysteme (Empfänger und Sender) gesendet werden, sollten ein *POLICY*-Objekt mit *AUTH_USER* verwenden, damit der Benutzer des Endsystems sich gegenüber der Firewall ausweisen kann.

Durch dieses Verfahren kann erreicht werden, dass eine Konfigurationsänderung (z.B. die dynamische Integration einer Filterregel) von FW_A und FW_B zum einen jeweils nur aus der eigenen

Domäne erfolgen kann und zum anderen sicher erfolgt.

Im obigen Beispiel ist damit für die Konfiguration von FW_A für den UDP-Strom die von S generierte und gesicherte *PATH*-Nachricht von entscheidender Bedeutung; für die Konfiguration von FW_B die gesicherte *RESV*-Nachricht von E.

Session Teardown. Wird der im Beispiel verwendete UDP-Strom nicht mehr benötigt (bzw. es soll seine Verwendung nicht mehr möglich sein), so muss der dafür freigeschaltete Weg innerhalb der Firewall-Systeme FW_A und FW_B wieder geschlossen werden. Dafür können folgende Mechanismen verwendet werden:

- **Sender-initiiertes Teardown:** Wird die Applikation, die den UDP-Strom verwendet, auf Seite des Senders beendet, so schickt der Sender eine *PATTEAR*-Nachricht an den Empfänger. Diese RSVP-Nachricht wird entlang des Pfades verschickt und kann durch die RSVP-Systeme verwendet werden, um eine zuvor durchgeführte Reservierung explizit zu löschen. In dem oben beschriebenen Beispiel kann dann in FW_A nach Erhalt der Nachricht die Konfigurationsänderung durchgeführt werden. Für FW_B ist dies nicht möglich, da die Nachricht ungesichert aus dem unsicheren Internet erhalten wird. Eine Konfigurationsänderung der FW_B darf nur aufgrund einer gesicherten und über das interne Interface erhalte Nachricht erfolgen. Dieses Problem kann umgangen werden, indem auf den Timeout des *path state* in FW_B gewartet wird. Alternativ besteht eventuell die Möglichkeit, dem Empfänger des UDP-Stroms über die Signalisierung der Applikation mitzuteilen, dass der UDP-Strom nicht mehr verwendet wird, so dass dieser seinerseits eine *RESVTEAR*-Nachricht generiert, die dann verwendet werden kann den *reservation state* in FW_B zu löschen.
- **Empfänger-initiiertes Teardown:** Geht das Aufheben der Reservierung vom Empfänger aus, so kann dieser die Reservierung durch Senden einer *RESVTEAR*-Nachricht aufheben. Diese Nachricht kann nur durch FW_B akzeptiert werden. Für FW_A kann wiederum ein Timeout oder eine *PATHTEAR*-Nachricht des Senders verwendet werden.
- **Firewall-initiiertes Teardown:** Wird eine Reservierung (respektive Firewall-Konfiguration) durch die Firewall selbst aufgehoben, so geschieht dies, indem die Firewall eine *PATHERR* bzw. *RESVERR* generiert. Entscheidet beispielsweise FW_A die Reservierung aufzuheben, wird eine *PATHERR*-Nachricht an den Sender geschickt. Der Sender kann daraufhin, wie oben beschrieben, einen Teardown initiieren.
- **Timer-initiiertes Teardown:** Jede Reservierung kann aufgehoben werden, indem diese nicht durch entsprechende Nachrichten aktualisiert wird.

Fehlerbehandlung. Es ist möglich, dass eine Reservierung (bzw. Konfigurationsanpassung) innerhalb einer Firewall nicht vorgenommen werden kann. Dafür kann es verschiedene Gründe geben; mit dem dadurch entstehenden Fehler muss entsprechend umgegangen werden:

- Wird beispielsweise die durch den Sender S geschickte *PATH*-Nachricht durch FW_A nicht akzeptiert, z.B. weil die Firewall für den angegebenen Host keine Kommunika-

tion mit der Außenwelt zulassen soll, so wird eine *PATHERR*-Nachricht generiert und an den Sender (über R_1) zurückgeschickt. Die *PATH*-Nachricht wird dabei nicht an R_2 weitergegeben.

- Wird die *RESV*-Nachricht des Empfängers durch FW_B abgelehnt, so wird eine *REVERR*-Nachricht an den Empfänger übermittelt.

Verwendung einer H.323-Applikation. Im Folgenden wird am Beispiel eines direkten Rufes zwischen zwei H.323-Terminals (mit *H.245tunneling*) die Funktionsweise der Firewall-Steuerung anhand eines Multimedia-Protokolls erläutert. Dazu wird wieder das Referenzszenario in Abbildung 24 verwendet, Sender (Terminal A) und Empfänger (Terminal B) stellen in diesem Fall H.323-Terminals dar. Des weiteren wird zur Vereinfachung angenommen, dass R_1, R_2 und R_3 gewöhnliche, nicht RSVP-fähige, Router sind. Die Endsysteme sowie die Firewall-Systeme verfügen über eine RSVP-Implementierung. Abbildung 29 zeigt den Ablauf der Kommunikation entsprechend den in Kapitel 3 gegebenen Definitionen der Darstellung:

- Terminal A muss zu Beginn der Kommunikation den Pfad für die Signalisierungsverbindung (Q.931+ H.245) anmelden. Dazu wird eine *PATH*-Nachricht an Terminal B verschickt. Diese enthält das für die Signalisierungsverbindung verwendete Protokoll (TCP), den verwendeten Zielport (Well-Known-Port 1720), die Zieladresse (IP_{TB1}), den Quellport (P_{TA1}) und die Quelladresse (IP_{TA1}).
- Diese *PATH*-Nachricht wird über die beteiligten RSVP-Knoten weitergeleitet und dort, wie oben beschrieben, verarbeitet.

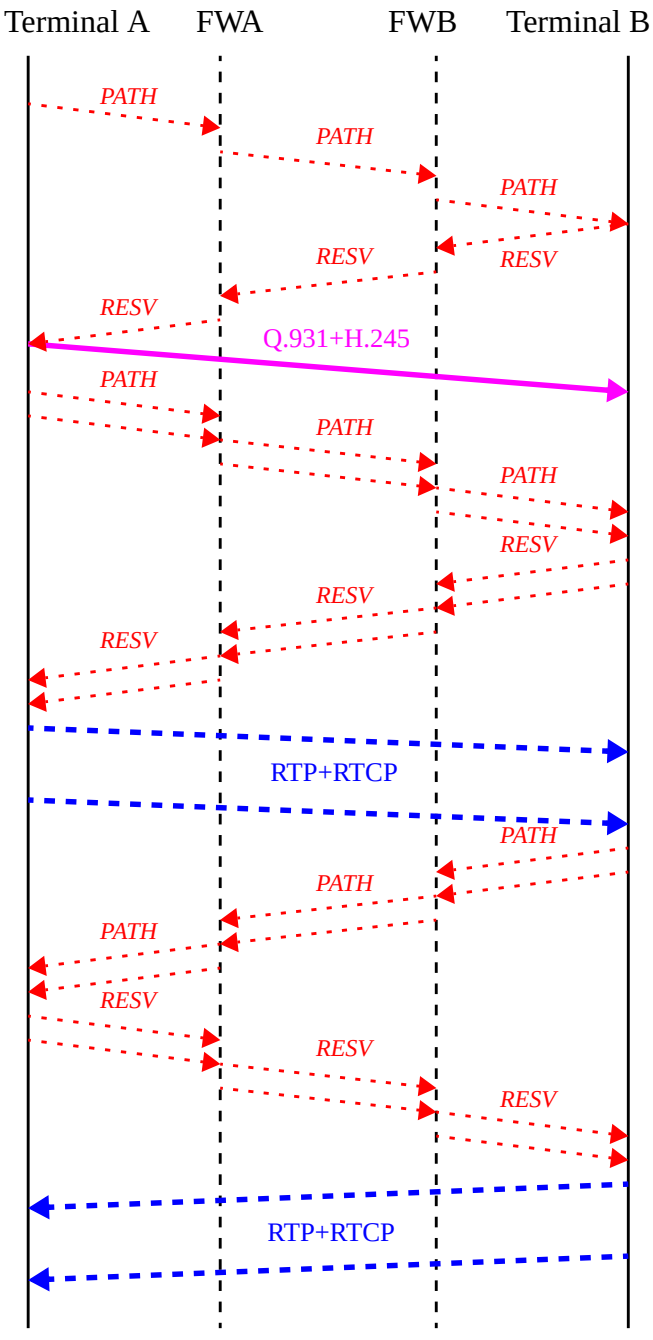


Abbildung 28: H.323/RSVP Interaktion (a)

- Nachdem Terminal B diese Nachricht erhalten hat, antwortet es mit einer *RESV*-Nachricht.
- Nach Erhalt der *RESV*-Nachricht durch FW_B wird das 5-Tupel (P_{TA1},IP_{TA1},1720, IP_{TB1},TCP) verwendet, um den Signalisierungspfad in FW_B freizuschalten. Es wird angenommen, dass eine bidirektionale Kommunikation durch die Firewall freigeschaltet wird. (Alternativ könnten zwei Reservierungen durchgeführt werden.)
- Danach kann der Signalisierungskanal durch Terminal A aufgebaut werden. Es werden die nötigen Signalisierungsinformationen zwischen den Terminals ausgetauscht, die zur Aushandlung der Parameter der Medienkanäle benötigt werden.
- Terminal A kann nun, da die Parameter der Medienströme bekannt sind, die verwendeten RTP- und RTCP-Ströme ankündigen. Dazu werden zwei *PATH*-Nachrichten an Terminal B verschickt.

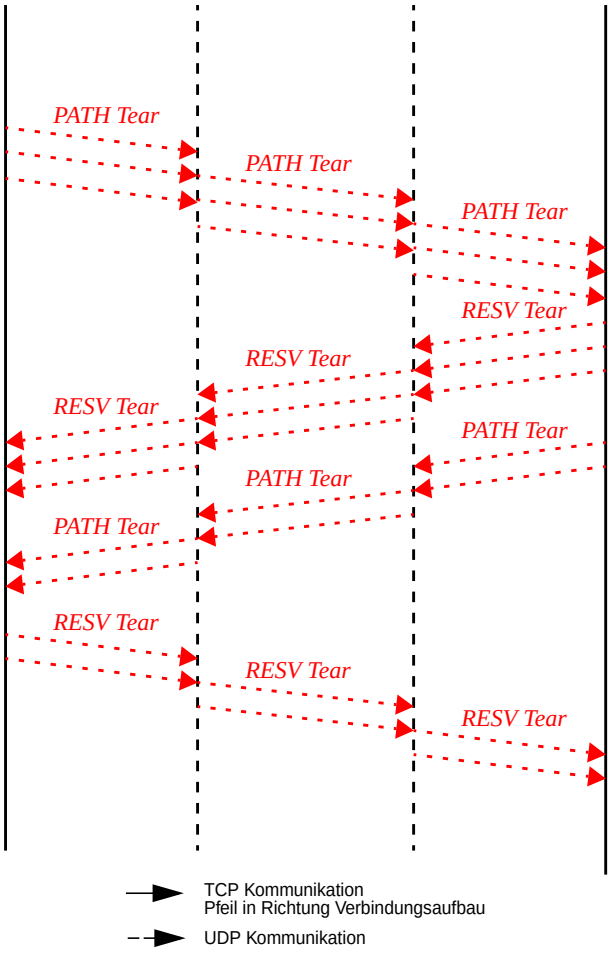


Abbildung 29: H.323/RSVP Interaktion (b)

- Diese *PATH*-Nachrichten werden entsprechend verarbeitet. Nachdem Terminal B diese erhalten hat werden passende *RESV*-Nachrichten generiert.
- Die *RESV*-Nachrichten werden wie beschrieben durch die Firewalls verwendet, um die Konfiguration für diese UDP-Ströme anzupassen.
- Für die von Terminal B verwendeten Ströme wird die selbe Reservierung entsprechend seitenverkehrt durchgeführt.
- Die Sitzung ist nun zwischen beiden Terminals aufgebaut. Während dieser Zeit müssen für alle Reservierungen periodisch *PATH*- und *RESV*-Nachrichten geschickt werden, damit die Reservierungen nicht durch ein Timeout entfernt werden.
- Wird die Sitzung beendet, werden die Firewall Konfigurationen durch *PATHTEAR*- und *RESV TEAR*-Nachrichten für jeden verwendeten Strom wieder aufgehoben. (Alternativ kann auch auf den Timeout gewartet werden)

Damit ist aufgezeigt, dass die beschriebene RSVP Firewall-Steuerung für Multimedia-Applikationen verwendet werden kann. In [102] sind weitere Beispiele für andere Multimedia-Applikationen gegeben.

5.5.2 Implementierung eines RSVP-FCP

In den vorangegangenen Abschnitten wurde gezeigt, wie RSVP theoretisch als Firewall Control Protocol verwendet werden kann. Im Folgenden wird eine technische Umsetzung des bisher beschriebenen Konzeptes dargestellt. Der Schwerpunkt liegt dabei auf der Integration einer Firewall und einer RSVP Implementierung (Firewall mit entsprechender *Firewall Control*). Als Endsysteme können normale RSVP-fähige Klienten verwendet werden.

Ansatz. In einem QoS-fähigen, über RSVP angesteuerten Router werden unter anderem ein *Packet Classifier* und ein *Packet Scheduler* verwendet. Beide Elemente werden über den im Router vorhandenen RSVP-Prozess angesteuert. Der *Packet Classifier* hat die Aufgabe, eingehende Pakete anhand des 5-Tupels den zuvor durch Reservierungen spezifizierten Strömen zuzuordnen. Der *Packet Scheduler* sorgt dann dafür, dass diese Pakete mit den spezifizierten QoS-Anforderungen übereinstimmen. Können Pakete nicht mehr an die QoS-Spezifikation angepasst werden, können diese beispielsweise verworfen werden. Vergleicht man die Aufgaben des *Packet Classifier* und des *Packet Scheduler* mit den in Kapitel 2 in Abbildung 2 beschriebenen Elementen einer Firewall, so ergeben sich wesentliche Übereinstimmungen. Der Analysator der Firewall entspricht dem *Packet Classifier*, Entscheider und Umsetzer zusammen entsprechen dem *Packet Scheduler*. Wird eine neue Reservierung durch den RSVP-Prozess an *Packet Classifier* und *Packet Scheduler* übermittelt, entspricht dies im Modell der Firewall einem Hinzufügen einer neuen Regel innerhalb des Regelspeichers. Zur Anbindung einer Firewall an einen RSVP-Prozess ist demnach die Schnittstelle für die Ansteuerung des *Packet Classifiers* und des *Packet Schedulers* innerhalb des RSVP-Prozesses zu verwenden.

Umsetzung. Zur Realisierung wurden zwei existierende Software-Pakete verwendet: die unter [103] verfügbare *KOM RSVP Engine* (in C++), sowie die unter [104] verfügbare Implementierung des Paketfilters *ip-filter* (in C). Beide Komponenten werden auf einem FreeBSD-System verwendet. Die RSVP Implementierung verwendet eine gemeinsame Schnittstelle zur Ansteuerung des *Packet Classifiers* und des *Packet Schedulers*. Diese Schnittstelle wird durch die Klasse *Scheduler* repräsentiert:

```
CLASS SCHEDULER {
 ADDFLOWSPEC();
 MODFLOWSPEC();
 DELFLOWSPEC();
 ADDFILTER();
 DELFILTER();
}
```

Es existieren verschiedene Klassen, die von der Basisklasse *Scheduler* erben und zur Ansteuerung verschiedener technischer Umsetzungen von *Classifier/Scheduler* dienen. Es kann über eine

Klasse der in [105] beschriebene ALTQ angesteuert werden. Über einen *ioctl* Funktionsaufruf kann der im User-Space ablaufende RSVP-Prozess dann bei Verwendung der Schnittstelle (z.B. *ADDFLOWSPEC()*) mit den im Kernel des Systems laufenden ALTQ interagieren. Um nun mit dem Paketfilter zu interagieren, wurde eine neue *Scheduler*-Klasse implementiert: *SchedulerFW.* Bei Verwendung des Interface (z.B. *ADDFLOWSPEC()*) wird über die durch den Filter angebotene *ioctl* Schnittstelle die entsprechende Information an den Paketfilter übermittelt. Wird beispielsweise *ADDFLOWSPEC()* verwendet, so wird aus dem *SESSION*-Objekt und dem *SENDER_TEMPLATE* ein 5-Tupel gebildet. Dieses 5-Tupel wird verwendet, um die für den *ioctl* verwendete Datentruktur zu füllen, danach wird der *ioctl* ausgeführt und die Konfigurationsänderung der Firewall erfolgt.

Zusammenfassung. Wie gezeigt wurde, existieren nicht nur methodische Übereinstimmungen zwischen RSVP und einem Firewall Control Protocol. Diese Übereinstimmungen finden sich ebenfalls in existierenden Implementierungen wieder. Durch die Anpassung nur weniger Elemente ist es möglich bestehende RSVP und Firewall Implementierungen zusammenzubringen.

5.6 Weiterführende Aspekte eines RSVP-FCP

QoS-Firewall. Werden auf dem verwendeten Pfad die RSVP-Nachrichten nicht nur zur Steuerung einer Firewall verwendet, sondern zugleich im herkömmlichen Sinn zur Umsetzung von QoS-Anforderungen, so sollte auch die Firewall in der Lage sein, die QoS relevanten Elemente der RSVP-Signalisierung umzusetzen. Die Firewall sollte an der Aushandlung der QoS-Anforderungen teilnehmen und dafür sorgen, dass diese innerhalb der Firewall umgesetzt werden. Dies ist insbesondere deshalb von Bedeutung, da eine Firewall in der Praxis oft den Flaschenhals auf einer Kommunikationsstrecke darstellt.

NAT. Die Verwendung von NAT-Komponenten innerhalb einer Firewall wurde bisher in diesem Kapitel nicht betrachtet. Lässt man die Verwendung von NAT-Komponenten ebenfalls zu, so ergeben sich zwei zu beachtende Änderungen:

- **NAT-OUT:** Nimmt man an, dass in Abbildung 24 FW_A über eine NAT-Komponente verfügt, so kann die durch S generierte *PATH*-Nachricht wie bisher an E gesendet werden. Wird die *PATH*-Nachricht durch FW_A erhalten, wird neben der Konfiguration des Filters ebenfalls die Konfiguration der NAT-Komponente vorgenommen. Für die Senderspezifikation (*Sender Template* (P_S,IP_S)) wird in FW_A eine Adress/Port-Umsetung angelegt (P_S<->P_S^I und IP_S<->IP_S^I). Diese Umsetzung muss bei der Weiterleitung der *PATH*-Nachricht innerhalb des *Sender Template* (dort werden nun P_S^I und IP_S^I verwendet) berücksichtigt werden. Für eine als Antwort eintreffende *RESV*-Nachricht muss dieses Mapping ebenfalls durchgeführt werden.
- **NAT-IN:** Verwendet FW_B eine NAT-Komponente, so ergibt sich das Problem, dass der Empfänger des Stroms nicht direkt adressiert werden kann, da dieser bei NAT in der Regel keine global gültige IP-Adresse besitzt. Der Empfänger muss dafür bei FW_B eine Adress/Port-Umsetzung anfragen (P_E<->P_E^I und IP_E<->IP_E^I). Eine solche Anfrage kann mit Hilfe des RSVP-Protokolls an FW_B gestellt werden, indem dafür entsprechende Nachrichten definiert werden. Über Standard-RSVP-Mechanismen kann das Problem *NAT-IN* jedoch nicht gelöst werden.

Damit ist gezeigt, dass auch NAT-Komponenten über RSVP angesteuert werden können, jedoch ist dafür die Definition neuer RSVP-Nachrichten notwendig.

5.7 Zusammenfassung

In den vorangegangenen Abschnitten wurde gezeigt, dass RSVP zur Umsetzung des *signalling part* des in Kapitel 4 beschriebenen Firewall-Modells geeignet ist. Es wurde gezeigt, dass RSVP sowohl in der Theorie als auch in der Praxis als Firewall Control Protocol verwendet werden kann. Aus der Verwendung von RSVP zur Realisierung einer Firewall ergeben sich verschiedene Vorteile. Wie gezeigt wurde, ist es möglich, wesentliche, zur Realisierung der Firewall notwendige Komponenten aus einer bestehenden RSVP Infrastruktur zu übernehmen. Zusätzlich werden verschiedene Probleme gelöst, die bei anderen Realisierungsvorschlägen auftreten, wie zum Beispiel das Auffinden von zu steuernden Firewall-Komponenten oder die Unterstützung von IPv6 Netzen. Im Umfeld von Multimedia-Applikationen bietet die Verknüpfung von RSVP und Firewall-Architektur darüber hinaus die Möglichkeit, Firewall-Systeme effizient in QoS-Architekturen zu integrieren.

Die in diesem Kapitel beschriebene Methode, RSVP als Firewall Control Protocol zu verwenden, wurde in [106] beschrieben und wurde mittlerweile in modifizierter Form als Vorschlag innerhalb der IETF (*Midcom*, [107]) eingebracht.

Kapitel 6: Signalisierungsverarbeitung in Firewall-Architekturen

In Kapitel 4 wurden verschiedene Firewall-Architekturen vorgestellt. Für jede Firewall-Architektur ist dabei von zentraler Bedeutung, wie die Signalisierungsverarbeitung realisiert werden kann. In diesem Kapitel werden auftretende Probleme bei der Umsetzung der Signalisierungsverarbeitung beschrieben und passende Lösungen für die auftretenden Probleme gegeben.

6.1 Motivation und Zielsetzung

Entsprechend des in Kapitel 3 festgelegten Designprinzips *Trennung von Signalisierungs- und Medienpfad* sollte die Signalisierungsverarbeitung getrennt von der Medienverarbeitung erfolgen. Dies erlaubt es, die Signalisierungsverarbeitung getrennt von der Medienverarbeitung zu betrachten. Wie in Kapitel 4 beschrieben, ist die Umsetzung der Signalisierungsverarbeitung in bestehenden Firewall-Systemen für die Klasse der Multimedia-Applikationen problematisch. Diese Problematik kann in die zu betrachtenden Teilaspekte “Integration” und “Verarbeitung” gegliedert werden. Ziel dieses Kapitels ist es, für beide Aspekte passende Lösungsansätze aufzuzeigen.

Integration. Das erste bei der Umsetzung der Signalisierungsverarbeitung für Multimedia-Applikationen auftretende Problem ist die Integration. Damit eine Firewall ihre Aufgabe erfüllen kann, muss diese in das zu unterstützende Kommunikationsszenario eingebunden werden. Dies ist zum einen problematisch, da die meisten Protokollstandards keine Firewall, und damit auch nicht das Einbringen einer solchen in den Signalisierungsweg, vorsehen. Zum anderen können, wie später gezeigt wird, gängige Integrationsverfahren nicht angewendet werden. Ziel ist es, zunächst zu untersuchen, warum bestehende Integrationsmechanismen nicht verwendet werden können. Danach werden davon ausgehend entsprechende Ansätze entwickelt, die eine effiziente Integration ermöglichen.

Verarbeitung. Kann eine Integration der Signalisierungsverarbeitung durchgeführt werden, so ist festzulegen, wie die eigentliche Verarbeitung des Signalisierungspfades durchgeführt werden muss. Die Verarbeitungslogik muss zunächst mit den in Kapitel 3 beschriebenen Charakteristika einer Multimedia-Applikation umgehen können. Dabei ist es das Ziel zu ermitteln, welche Verarbeitungsmechanismen dafür notwendig sind. Des Weiteren ist zu betrachten, welche Verarbeitungsmechanismen nötig sind, um die durch eine Firewall zu erbringenden Schutzfunktionen umsetzen zu können.

Für beide zu betrachtenden Aspekte wird durch eine Implementierung gezeigt, dass die vorgeschlagenen Lösungen auch in der Praxis eine Lösung der Probleme darstellen. Der Fokus liegt dabei auf H.323-Szenarien, die als Beispiel-Szenarien verwendet werden.

6.2 Untersuchung der Integration einer Signalisierungsverarbeitung

Im Folgenden wird beschrieben, welche klassischen Integrationsmethoden für eine Firewall zur Zeit verwendet werden. Danach wird anhand eines Telefonieszenarios basierend auf H.323 gezeigt, dass sich die klassischen Integrationsmethoden nicht anwenden lassen, um alle notwendigen Szenarien zu unterstützen. Daran anschließend wird aufgezeigt, dass in einem H.323-Szenario weitere Integrationsmöglichkeiten bestehen, die es ermöglichen die fehlenden Szenarien zu unterstützen. Es wird darüber hinaus dargelegt, dass sich diese Integrationsmethoden auch für andere Applikationsszenarien als H.323-Szenarien verwenden lassen.

6.2.1 Klassische Integrationsmethoden

Klassische Firewall-Architekturen, wie beispielsweise ein Hybridsystem, verwenden in der Regel entweder eine transparentes oder explizites Integrationsverfahren. Zur Erläuterung der verschiedenen Integrationsmethoden wird das in Abbildung 30 dargestellte Szenario verwendet:

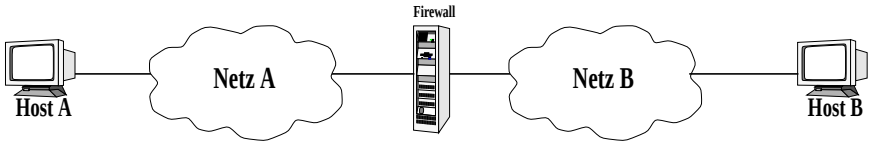


Abbildung 30: Integration der Signalisierungsverarbeitung

Transparente Integration. Bei einer transparenten Integration müssen die durch eine Firewall getrennten Kommunikationspartner nicht wissen, dass eine Firewall existiert. Die Firewall wird transparent in den Kommunikationsweg eingebunden. Wird angenommen, dass Host A eine TCP-Verbindung zu Host B aufbauen möchte (z.B. um eine HTTP-Verbindung zu etablieren), verwendet Host A zur Adressierung die IP-Adresse IP_B und den Port P_B des gewünschten Dienstes von Host B. Die verwendete Adressierung unterscheidet sich damit nicht von einer Adressierung, bei der keine Firewall innerhalb des Kommunikationsweges verwendet wird. Die Firewall bringt sich nun beispielsweise in die Kommunikation ein, indem die Verbindung (identifiziert durch das 5-Tupel) an einen Analyseprozess innerhalb der Firewall umgelenkt wird. Dies wird erreicht, indem Zieladresse und Zielport durch eine Funktion innerhalb der Firewall umgeschrieben werden (IP_B<->IP_{FW}, P_B<->P_{FW}). Die Firewall kann die nun umgelenkte Verbindung ihrerseits komplettieren, indem der Analyseprozess innerhalb der Firewall seinerseits Host B unter IP_B und P_B kontaktiert. Damit wurde in diesem Beispiel die TCP-Verbindung transparent aufgetrennt und ein Analyseprozess in den Kommunikationsweg eingebracht.

Für das transparente Umlenken bzw. Einbringen der Firewall in den Kommunikationsweg gibt es verschiedene technische Umsetzungen, die aber alle das eben beschriebene Resultat erzielen. Für die Firewall besteht der Vorteil dieses beschriebenen Verfahrens darin, dass das Umlenken der Verbindung auf Schicht 3 und 4 erfolgt und damit nicht applikationsspezifisch ist. Es muss somit

für verschiedene Applikationen jeweils nur der Applikationsprozess (Protokoll-Parser) bei Auftreten neuer Applikationstypen angepasst werden. Nachteile dieses Verfahrens bestehen dann, wenn die Firewall eine NAT-Komponente enthält. Verwendet Netz B keine öffentlich gültigen IP-Adressen, kann Host A Host B nicht direkt adressieren, da dieser keine öffentlich gültige Adresse besitzt. Ein solches Szenario kann durch das transparente Integrationsverfahren nicht unterstützt werden.

Explizite Integration. Im Gegensatz zur transparenten Integration müssen bei der expliziten Integration die Kommunikationspartner (bzw. der Initiator einer Verbindung) von der Existenz der Firewall wissen. Der Initiator einer Verbindung adressiert explizit zuerst die Firewall und teilt dieser nachfolgend mit, welches der zu kontaktierende Kommunikationsendpunkt ist. Nimmt man wieder an, dass Host A eine TCP-Verbindung zu Host B aufbauen möchte, verwendet Host A zur Adressierung die IP-Adresse IP_{FW} und den Port P_{FW} des Applikationsprozesses der Firewall. Nachdem diese Verbindung aufgebaut ist, muss der Firewall explizit mitgeteilt werden, welches Endsystem kontaktiert werden soll, um die Verbindung zu vervollständigen. Diese explizite Mitteilung kann Out-Of-Band (z.B. über eine statische Konfiguration der Firewall, oder ein separates Signalisierungsprotokoll) oder In-Band (z.B. bei HTTP durch Übermittlung des *GET*-Request in welchem die Zielinformation enthalten ist) erfolgen. Nachdem die Informationen über das Endsystem mitgeteilt wurden (IP_B und P_B), wird dieses durch die Firewall kontaktiert. Die Verbindung ist nun vollständig aufgebaut und die übermittelten Daten können durch den zwischengeschalteten Analyseprozess interpretiert werden.

Vorteil der expliziten Methode ist, dass auch in einem NAT Szenario das Ziel adressiert werden kann (vorausgesetzt der Initiator der Verbindung kann eine Zielspezifikation an die Firewall übermitteln, die von dieser interpretiert werden kann). Nachteil dieses Verfahrens ist die Tatsache, dass die eingesetzten Endsysteme in der Lage sein müssen explizit mit einer Firewall zu interagieren. Verwendet man eine In-Band Signalisierung ergibt sich zusätzlich das Problem, die Zielinformationen innerhalb des auf der Verbindung verwendeten Applikationsprotokolls unterbringen zu müssen. Das Einbringen beliebiger Informationen wird aber in der Regel durch die verwendeten Protokolle nicht zugelassen.

6.2.2 Anwendung der klassischen Integrationsmethoden in H.323-Szenarien

Im Folgenden wird nun untersucht, wie die zuvor beschriebenen klassischen Integrationsmethoden in einem H.323-Szenario verwendet werden können. Das in Abbildung 31 dargestellte Szena-

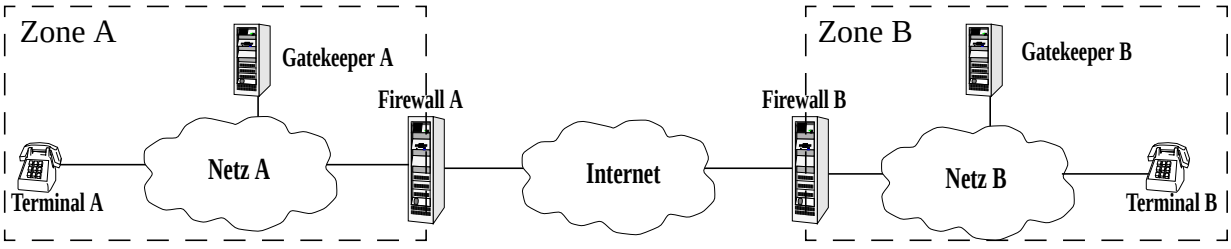


Abbildung 31: Integration der Signalisierungsverarbeitung in einem H.323-Szenario

rio wird für die Erläuterung verwendet. Innerhalb des Szenarios sind zwei Netze gezeigt, die über

das Internet miteinander verbunden sind. Beide Netze werden über die dazugehörende Firewall gegen Angriffe aus dem Internet geschützt.

Transparent - Firewall Redirect. Im folgenden Beispiel wird zunächst angenommen, dass in dem in Abbildung 31 gegebenen Szenario eine direkte Kommunikation (ohne Einbeziehung der Gatekeeper) zwischen Terminal A und Terminal B stattfindet. Außerdem wird angenommen, dass Firewall B durch einen einfachen Router ersetzt wird. In diesem Fall befindet sich zwischen Terminal A und Terminal B nur Firewall A. Es wird weiterhin angenommen, dass die Umlenkung über ein Umschreiben der IP-Adresse und Portnummer geschieht (*Redirect*). Wird der direkte Ruf zwischen A und B durch Terminal A initiiert ergibt sich folgender Kommunikationsablauf:

- **Voraussetzung:** Die Firewall ist in der Lage bestimmte Ströme, identifizierbar an Protokoll, Ziel-Port, Ziel-IP-Adresse, Quell-Port und Quell-IP-Adresse, umzuleiten. Dies geschieht, indem die Firewall die IP-Adressen und Port-Nummern der einzelnen Pakete eines Stroms modifiziert und den Zustand der so umgeleiteten Ströme in einer Tabelle (*Redirect Table*) speichert.
- **Q.931 Call Signalling (TCP):** Bei einem Verbindungsaufbau kontaktiert das Terminal A direkt Terminal B. Die Firewall verwendet ihre Redirect-Fähigkeit, um den Q.931-Strom transparent umzuleiten. Dabei wird der Q.931-Strom an die Firewall selbst umgelenkt, wodurch diese in die Signalisierung eingebunden wird. Um nun das eigentliche Ziel (Terminal B) zu bestimmen und dann zu kontaktieren, stehen der Firewall zwei Möglichkeiten zur Verfügung. Erstens kann die *Redirect Table* der Firewall verwendet werden, um das Ziel zu bestimmen, da dort die Zuordnung zwischen Quelle (Terminal A) und eigentlichem Ziel (Terminal B) festgehalten ist. Zweitens kann die *Setup*-Nachricht analysiert werden, die Informationen über das Ziel enthält. Nach Bestimmung des Ziels kann die Firewall das Terminal B kontaktieren und die jeweils auftretenden Q.931-Nachrichten zum jeweiligen Terminal weiterleiten.

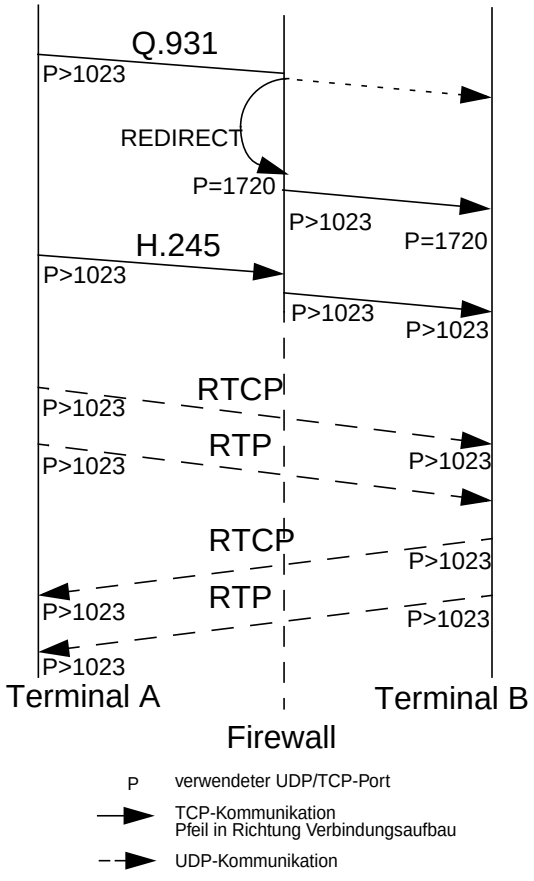


Abbildung 32: Firewall-Redirect

- **H.245 Call Control (TCP), RTP/RTCP Media und Mediacontrol (UDP):** Nachdem die Firewall nun in die Q.931-Verbindung integriert ist, kann sie den dort ausgehandelten Parameter (Port und IP-Adresse) des H.245-Kanals mitbekommen und ändern. Dadurch ist die Firewall in der Lage ebenfalls den zweiten Signalisierungskanal zu kontrollieren. Damit kann

die Firewall nun auch Kenntnis über die ausgehandelten Medienkanäle erlangen und die Firewall entsprechend konfigurieren.

Diese Methode ist transparent, das Terminal A adressiert das Terminal B in gewohnter Weise. Die Umlenkung des initialen Q.931-TCP-Stroms ist für das Terminal A nicht erkennbar und damit transparent. Allerdings kann diese Methode nur dann verwendet werden, wenn Terminal B durch Terminal A direkt adressiert werden kann. Aus diesem Grund kann dieses Verfahren nicht in NAT-Umgebungen für eingehende Rufe verwendet werden.

Explizit - Modifikation von H.323-Komponenten. In einem H.323-Szenario zeichnet sich die explizite Variante dadurch aus, dass eine oder mehrere H.323-Komponenten (z.B. ein Terminal) so verändert werden, dass die Firewall in den Kommunikationsweg eingebracht wird. Dies geschieht dann nicht transparent, sondern explizit. Legt man das gleiche Kommunikationsbeispiel wie bei der zuvor beschriebenen transparenten Methode zugrunde, so ergibt sich folgender Kommunikationsablauf:

- **Voraussetzung:** Das externe Terminal A muss modifiziert werden. Es ist eine Konfigurationsmöglichkeit vorzusehen, die es dem Benutzer erlaubt, die Adresse der Firewall (zusätzlich zum Rufziel) einzugeben. Die so spezifizierte Firewall kann dann den Ruf zwischen externem und internem Netz vermitteln.
- **Q.931 Call Signalling (TCP):** Bei einem Verbindungsaufbau kontaktiert das Terminal A zunächst die Firewall. In diesem Fall wird die Q.931-Verbindung von Terminal A zur Firewall aufgebaut. Innerhalb der Q.931-Verbindung wird dann als erstes eine *Setup*-Nachricht von Terminal A an die Firewall übermittelt. Innerhalb dieser Nachricht ist angegeben, welches die eigentliche Zieladresse des Rufes ist. Die Zieladresse kann dabei in verschiedenen Formaten angegeben sein, beispielsweise als E.164-Nummer, als IP-Adresse, als H323-ID oder in einer anderen Form. Die Firewall muss nun in der Lage sein, diese Zieladresse in eine IP-Adresse umzusetzen. Danach kann die Firewall das Terminal B kontaktieren und die jeweils auftretenden Q.931-Nachrichten zum jeweiligen Terminal weiterleiten.

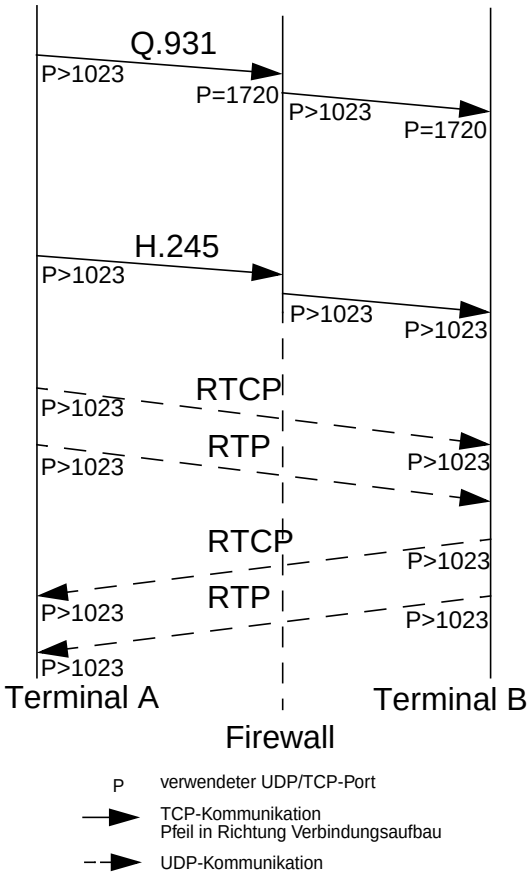


Abbildung 33: Explizite Signalisierung

- **H.245 Call Control (TCP), RTP/RTCP Media und Mediacontrol (UDP):** Analog zu den im vorhergehenden Abschnitt beschriebenen Mechanismen kann die Firewall sich nun in die

gesamte Kommunikation einschalten, indem die übermittelten Kanalspezifikationen in den Signalisierungsnachrichten modifiziert werden.

Diese Methode ist nicht transparent, das Terminal muss verändert werden. Rufe, die über die Firewall vermittelt werden, werden explizit anders behandelt als direkte Rufe zwischen zwei Terminals. Ein Beispiel, das diese Form des Routing über eine Firewall unterstützt, ist die Microsoft Netmeeting Terminalsoftware [108]. Diese Terminalsoftware besitzt die Möglichkeit, eine Firewall (dort als Gateway bezeichnet) für die ausgehenden Rufe anzugeben.

Schlussfolgerung. Wie beschrieben bergen in einem H.323-Szenario beide klassische Methoden Probleme. Die transparente Methode kann nicht verwendet werden, wenn ein Terminal in einem durch eine NAT-Komponente abgetrennten Bereich einen Ruf empfangen soll. Die explizite Methode besitzt den Nachteil, dass jedes Endgerät modifiziert werden muss, was in der Praxis meist nicht umzusetzten ist. Ein weiteres Problem tritt bei der expliziten Methode dann auf, wenn mehrere Firewalls auf dem Kommunikationsweg liegen. In diesem Fall müssen die notwendigen Routinginformationen an mehrere Firewalls übermittelt werden. Zusätzlich muss das Endsystem Wissen über die dem Szenario zugrunde liegende Topologie besitzen.

Konventionelle Applikationen (z.B. HTTP, FTP) sind meist Client-Server-basiert. Die Server besitzen eine öffentlich gültige IP-Adresse, der Client befindet sich hinter einer Firewall und initiiert die Verbindung (z.B. Zugriff auf einen WWW-Server). Für dieses Kommunikationsmodell können die beschriebenen Integrationsmethoden problemlos verwendet werden. Bei Multimedia-Applikation sind aber in der Regel beide Kommunikationsendpunkte gleichberechtigt. Beide Endpunkte befinden sich in ähnlich gesicherten Netzen und beide Endpunkte können die Verbindung initiieren. Aus diesem Grund eignen sich die klassischen Integrationsmethoden nicht für die Klasse der Multimedia-Applikationen.

6.2.3 Integrationsmethoden in H.323-Szenarien

Neben den klassischen Integrationsmethoden können in einem H.323-Szenario weitere Methoden verwendet werden, die es erlauben eine Firewall in den Kommunikationsweg einzubringen. Die im Folgenden dargestellten Methoden nutzen die Tatsache, dass innerhalb eines H.323-Szenarios verschiedene Infrastrukturkomponenten (Gatekeeper, Gateways, MCU) verwendet werden. Für diese Infrastrukturkomponenten sind innerhalb des H.323-Standards Methoden vorgesehen, diese in ein Kommunikationsszenario einzubinden. Durch Kombination einer solchen Infrastrukturkomponente mit einer Firewall kann das Integrationsproblem gelöst werden.

Infrastruktur - Nutzung eines Gatekeepers. Um die Firewall in die Kommunikation einzubinden, kann ein H.323-Gatekeeper in ein Firewall-System integriert werden. Wird dann ein *Gatekeeper Routed Call Model* verwendet (siehe Kapitel 3 Abschnitt 3.4), so ist die Firewall durch die Infrastrukturkomponente in die Kommunikation eingebunden. Legt man Abbildung 31 zugrunde, kann folgendes Szenario verwendet werden: Gatekeeper A fällt zusammen mit Firewall A, Terminal A verwendet Gatekeeper A (und damit die Firewall A), um Terminal B zu kontaktieren. Es wird hier wieder zunächst angenommen, dass Gatekeeper B nicht verwendet wird. Es ergibt

sich ein Kommunikationsablauf, der exakt dem in Kapitel 3 in Abbildung 13 beschriebenen Ablauf entspricht.

- **Voraussetzung:** Der Gatekeeper muss eine Interaktionsmöglichkeit mit der Firewall besitzen, um dieser die zur Erfüllung ihrer Aufgaben notwendigen Informationen zukommen lassen zu können.

Es ist ebenfalls möglich, das Szenario zu vergrößern und anzunehmen, dass Terminal B ebenfalls über seinen Gatekeeper B kommuniziert, der mit Firewall B zusammenfällt. In diesem Fall läuft die Kommunikation dann über zwei Gatekeeper (bzw. Firewalls). Dieses Verfahren ermöglicht es, die Firewall auf eine Art in die H.323-Kommunikation einzubringen, die sich sehr gut mit den H.323-Konzepten vereinbaren lässt. Innerhalb des H.323-Standards ist ein Gatekeeper als zentrales Element vorgesehen, eine Firewall aber nicht. Der Nachteil dieses Verfahrens ist, dass ein vollständiger Gatekeeper in eine Firewall integriert werden muss. Dies ist oft bei bestehenden Firewall-Systemen nicht möglich, zusätzlich entsteht dadurch ein enormes Sicherheitsproblem. Ein Gatekeeper kann selbst Ziel vieler Angriffe sein (siehe Kapitel 3 Abschnitt 3.4.2), außerdem werden viele Funktionen eines Gatekeepers nicht zwingend zur Erfüllung der Firewall-Funktionalitäten benötigt. Ein Beispiel, dass diese Form der Integration unterstützt, ist der Cisco MMCM [109]. Dieses Firewall-System besteht aus der Verbindung eines Gatekeepers mit einer Firewall.

Infrastruktur - Nutzung der Gatekeeper-Gatekeeper Kommunikation. Betrachtet man ein Szenario, in dem eine Kommunikation über verschiedene H.323-Zonen läuft, ergeben sich spezielle Möglichkeiten, eine Firewall in den Kommunikationsweg einzubringen. Es wird wieder das in Abbildung 31 gezeigte Szenario zugrunde gelegt. Terminal A ist dabei bei Gatekeeper A angemeldet, Terminal B bei Gatekeeper B. In dem dargestellten Beispiel wird ein Ruf, der von Terminal A nach Terminal B durchgeführt wird, durch beide dargestellten H.323-Zonen laufen. Die jeweiligen Zonengrenzen entsprechen den Bereichen, die durch die jeweilige Firewall geschützt werden. Dies entspricht in vielen Fällen der Realität, da die H.323-Zone sowie der durch die Firewall zu schützende Bereich von derselben administrativen Einheit betrieben wird. Im Folgenden wird zur Vereinfachung zunächst angenommen, dass nur Firewall A verwendet wird, Firewall B wird als gewöhnlicher Router angenommen. Es ergibt sich der folgende Kommunikationsablauf:

- **Voraussetzung:** Rufendes und gerufenes Terminal verwenden jeweils einen eigenen Gatekeeper. Die verwendeten Gatekeeper sind in der Lage über sog. RAS *LocationRequests* (*LRQ*) mit anderen Gatekeepern zu kommunizieren. Entsprechend benötigen die Gatekeeper eine Konfigurationsmöglichkeit, die ihnen angibt, welche *LRQ*s an welchen externen Gatekeeper

weitergeleitet werden müssen. Dies wird innerhalb der Gatekeeper in sog. *Neighbour Tables* festgelegt.

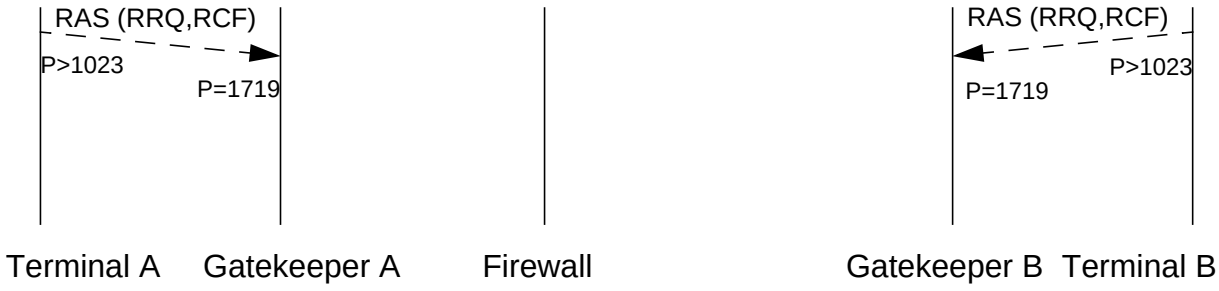


Abbildung 34: Gatekeeper-Gatekeeper Kommunikation (a)

- **RAS Registration, Admission, Status (UDP):** Nach dem Start der Terminals registrieren sich diese jeweils an dem ihnen zugewiesenen Gatekeeper (*RegistrationRequest RRQ* und *RegistrationConfirm RCF*).

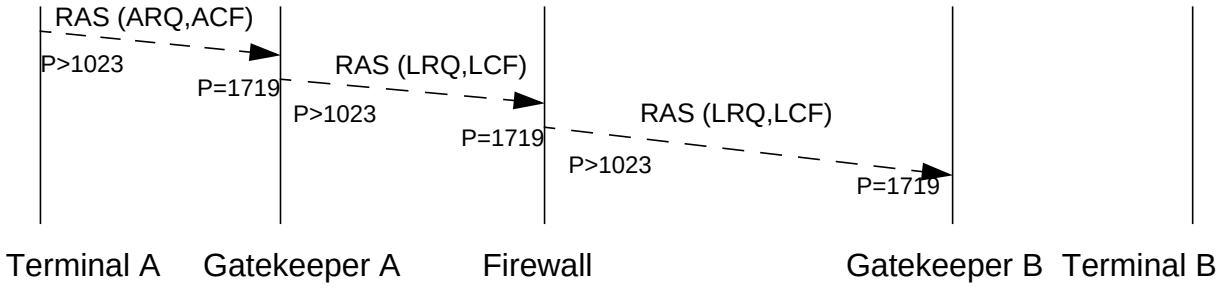


Abbildung 35: Gatekeeper-Gatekeeper Kommunikation (b)

- **RAS Registration, Admission, Status (UDP):** Bevor die Kommunikation stattfinden kann, muss das rufende Terminal beim Gatekeeper unter Verwendung des RAS-Protokolls mit einer Spezifikation der geplanten Gesprächscharakteristika eine entsprechende Erlaubnis erbitten (*AdmissionRequest ARQ*).
- **RAS Registration, Admission, Status (UDP):** In der *ARQ*-Nachricht ist die Zieladresse von Terminal B enthalten. Die Zieladresse kann dabei in verschiedenen Formaten angegeben sein, beispielsweise als E.164-Nummer, als IP-Adresse, als H323-ID oder in einer anderen Form. Es wird in diesem Beispiel angenommen, dass die Zieladresse von Terminal B als E.164-Nummer angegeben ist. Da kein Terminal unter dieser Nummer bei Gatekeeper A registriert ist, verwendet Gatekeeper A nun seine *Neighbour Table* um herauszufinden, ob ein erreichbarer Gatekeeper nach einem registrierten Terminal mit dieser Nummer gefragt werden kann. Diese Anfrage wird innerhalb einer *LRQ*-Nachricht an den entfernten Gatekeeper übermittelt. In diesem Beispiel existiert in der *Neighbour Table* ein einziger Eintrag: alle durch den Gatekeeper generierten *LRQ*-Anfragen werden an die Firewall weitergeleitet.
- **RAS Registration, Admission, Status (UDP):** Die Firewall erhält die *LRQ*-Nachricht. In der Nachricht sind die folgenden Werte enthalten: Eine Sequenznummer der Nachricht; die E.164-Nummer, zu der das entsprechende Terminal gesucht wird; Die IP-Adresse, an welche

die Antwort auf die *LRQ*-Nachricht gesendet werden soll. Die Firewall selbst benötigt nun ebenfalls eine *Neighbour Table*, um zu bestimmen, wohin die Anfrage weitergeleitet werden soll. Wir nehmen an, dass die Firewall anhand des Nummern Prefixes der übermittelten E.164-Nummer Gatekeeper B als Ziel der *LRQ*-Nachricht ermittelt. Bevor die *LRQ*-Nachricht von der Firewall an Gatekeeper B weitergeleitet wird, merkt sich die Firewall die original *LRQ*-Nachricht und ersetzt die enthaltene IP-Adresse, an die die Antwort gesendet werden soll, durch die IP-Adresse der Firewall. Dadurch wird Gatekeeper B veranlasst, die Antwort an die Firewall und nicht direkt an Gatekeeper A zu senden. Danach wird die *LRQ*-Nachricht an den Gatekeeper B weitergeleitet.

- **RAS Registration, Admission, Status (UDP):** Gatekeeper B erhält nun die *LRQ*-Nachricht. Da Terminal B unter der übermittelten E.164-Nummer angemeldet und damit erreichbar ist, sendet Gatekeeper B an die Firewall eine *LocationConfirm*-Nachricht (*LCF*) an die Firewall zurück. Diese Nachricht enthält folgende Informationen: Die Sequenznummer des *LRQ*, auf die sich das *LCF* bezieht; die Signalisierungszieladresse (IP-Adresse und Port für den folgenden Call Signalling Strom). Es wird in diesem Beispiel angenommen, dass Gatekeeper B einen nicht von einem Gatekeeper vermittelten Ruf verwenden möchte und deshalb als Call Signalling Adresse direkt die IP-Adresse des Terminals B übermittelt. Die Firewall benutzt nun die in der *LCF* und der zuvor gespeicherten *LRQ*-Nachricht enthaltenen Informationen, um eine Call-Routing Tabelle aufzubauen. Die Tabelle enthält jetzt für die E.164-Nummer die Ziel-IP-Adresse für das Call Signalling (in diesem Fall die IP-Adresse von Terminal B). Bevor die *LCF* an den Gatekeeper A weitergeleitet wird, ersetzt die Firewall die *Call Signalling* Adresse durch die eigene IP-Adresse.
- **RAS Registration, Admission, Status (UDP):** Gatekeeper A sendet nun auf die initiale *ARQ*-Anfrage des Terminals A eine *ACF*-Nachricht, die dem Terminal bestätigt, dass das gewünschte Ziel-Terminal erreichbar ist. Zusätzlich enthält die *ACF*-Nachricht die *Call Signalling* Adresse, welche für den Ruf verwendet werden soll. In diesem Beispiel nehmen wir an, dass Gatekeeper A einen von einem Gatekeeper vermittelten Ruf verlangt. Als *Call Signaling* Adresse wird deshalb die IP-Adresse von Gatekeeper A in der *LCF*-Nachricht angegeben.

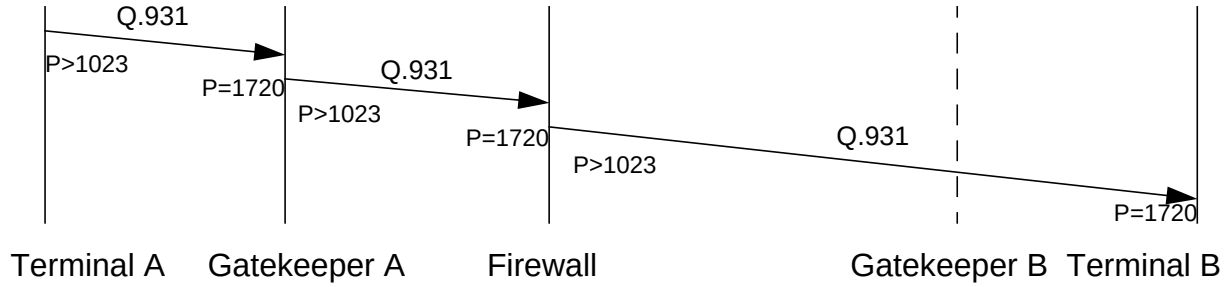


Abbildung 36: Gatekeeper-Gatekeeper Kommunikation (c)

- **Q.931 Call Signalling (TCP):** Bei einem Verbindungsaufbau kontaktiert das Terminal A nun, entsprechend der in der *ACF*-Nachricht übermittelten Parameter, Gatekeeper A. Danach wird die initiale *Setup*-Nachricht übermittelt, die die E.164-Adresse des gewünschten Ziels enthält.

- **Q.931 Call Signalling (TCP):** Gatekeeper A verwendet die übermittelte E.164-Adresse, um zu erkennen, dass das zu kontaktierende Ziel die Firewall ist. Diese Information wurde zuvor über die *LCF*-Nachricht dem Gatekeeper A mitgeteilt. Die *Setup*-Nachricht wird nun an die Firewall weitergeleitet.
- **Q.931 Call Signalling (TCP):** Die Firewall verwendet wiederum die E.164-Adresse, um das Ziel des Rufes zu verwenden. Wie zuvor beschrieben, wurden die *LRQ*- und *LCF*-Nachrichten dazu verwendet, eine Call-Routing Tabelle in der Firewall aufzubauen. Diese wird nun verwendet, um das Ziel zu bestimmen. Die Firewall kontaktiert daraufhin Terminal B und leitet die *Setup*-Nachricht an das Terminal weiter.
  Das Call Signalling ist nun vollständig aufgebaut, und die folgenden Q.931-Nachrichten können zwischen den Terminals über die jeweilig involvierten Infrastrukturkomponenten ausgetauscht werden.

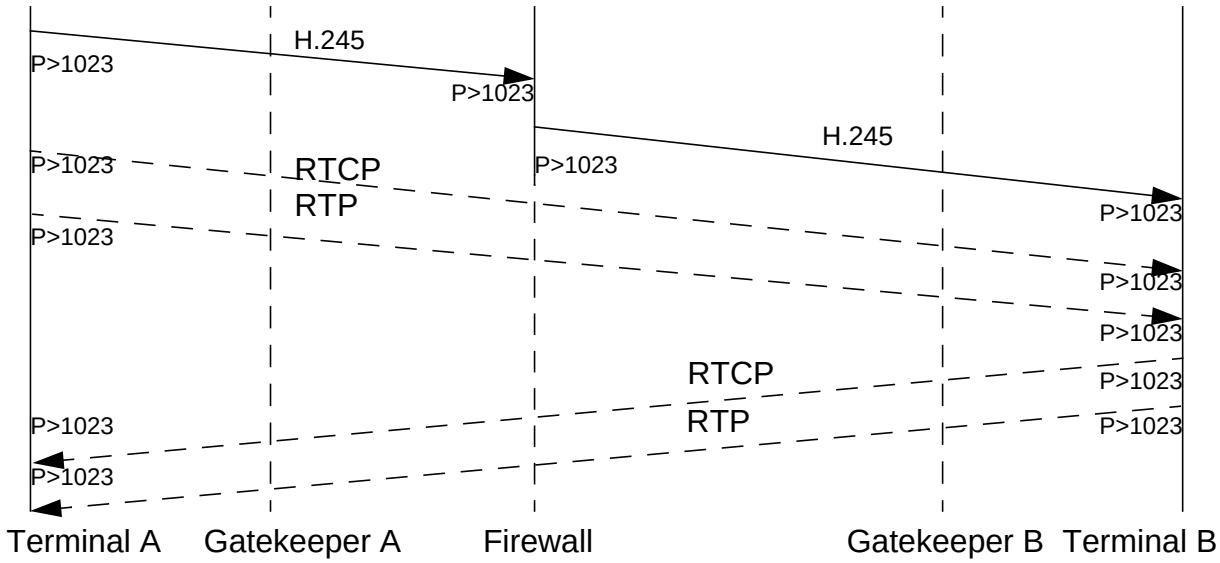


Abbildung 37: Gatekeeper-Gatekeeper Kommunikation (d)

- **H.245 Call Control (TCP), RTP/RTCP Media und Mediacontrol (UDP):** Analog zu den im vorigen Abschnitt beschriebenen Mechanismen kann die Firewall sich in die gesamte Kommunikation einschalten, indem die übermittelten Kanalspezifikationen in den Signalisierungsnachrichten modifiziert werden. In dem hier gegebenen Beispiel wird der H.245-Strom über die Firewall geleitet, nicht aber über die Gatekeeper. Die Medienströme fließen direkt zwischen den Terminals.

Dieser Mechanismus ist ebenfalls verwendbar, wenn auch in Zone B eine Firewall verwendet wird. Der Vorteil dieses Verfahrens besteht darin, dass zum einen die Firewall gut in die H.323-Szenarien eingebunden werden kann und zum anderen innerhalb der Firewall keine Implementierung eines vollständigen Gatekeepers nötig ist. Dadurch kann die Komplexität einer solchen Firewall reduziert werden, die Firewall muss nur die oben beschriebenen Mechanismen unterstüt-

zen. Die Firewall wird so zwar auch wie ein Gatekeeper in das Szenario eingebunden, die Firewall benötigt aber nicht die vollständige Gatekeeper-Implementierung.

6.2.4 Zusammenfassung

Wie gezeigt wurde, können mit den klassischen Integrationsverfahren nicht alle notwendigen Szenarien abgedeckt werden. Es wurde anhand eines H.323-Szenarios dargestellt, dass in einem Multimedia-Szenario weitere Integrationsmethoden zur Verfügung stehen, die sich die Funktionsweise von Infrastrukturkomponenten zu Nutze machen. Diese Integrationsmöglichkeiten können auch in anderen Applikationsszenarien angewendet werden. In einem SIP-Szenario kann eine Firewall entsprechend einem SIP-Proxy eingebunden werden. In einem RTSP-Szenario kann eine Firewall analog zu einem Proxy/Cache eingebunden werden.

Eine Infrastrukturkomponente kann auf verschiedene Arten für eine Integration genutzt werden. Damit die Komplexität einer Firewall gering gehalten werden kann und die notwendigen Sicherheitsmerkmale umgesetzt werden können, sollte nicht die Infrastrukturkomponente innerhalb der Firewall verwendet werden, sondern nur die notwendigen Mechanismen der Signalisierungsverarbeitung der Infrastrukturkomponente innerhalb der Firewall. Es kann somit folgendes Designprinzip formuliert werden:

> *Für die **Integration** einer Firewall in ein Multimedia-Applikationsszenario sollten vorhandene Mechanismen zur Integration von Infrastrukturkomponenten verwendet werden.*

Dieses Designprinzip wurde in [68], [110] und [111] beschrieben. In Abschnitt 6.4 wird gezeigt, dass bei Verwendung dieses Prinzips zur Zeit relevante Kommunikationsszenarien durch eine Firewall unterstützt werden können.

6.3 Anforderungen an eine Signalisierungsverarbeitung

Nachdem festgestellt wurde, dass es sich anbietet, eine Integration basierend auf Infrastrukturkomponenten zu verwenden, stellt sich die Frage, welche Verarbeitungsmechanismen für den Signalisierungspfad innerhalb der Firewall nötig sind. Um die notwendigen Verarbeitungsmechanismen identifizieren zu können, müssen folgende Aspekte betrachtet werden: Mechanismen für die Integration, Mechanismen, um mit den in Kapitel 3 beschriebenen Charakteristika einer Multimedia-Applikation umgehen zu können und Mechanismen, um die notwendigen Sicherheitsfunktionen umsetzen zu können. Im Folgenden wird betrachtet, wie sich die Berücksichtigung dieser drei Aspekte auf die Architektur der Signalisierungsverarbeitung auswirkt. Zur Erläuterung folgt eine Darstellung der Verarbeitungslogik:

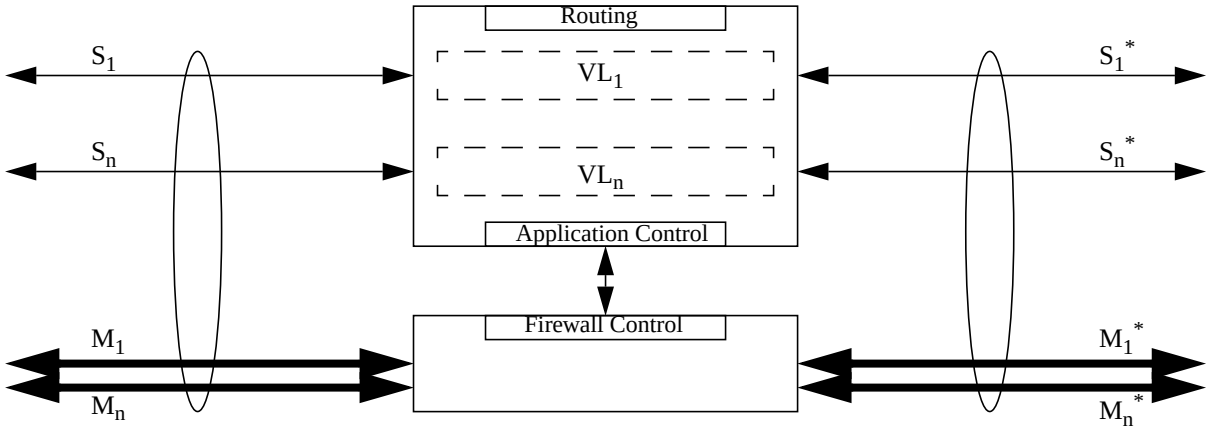


Abbildung 38: Verarbeitungslogik für den Signalisierungspfad

Integration. Verwendet man das oben beschriebene Designprinzip für die Umsetzung der Integration, so müssen dafür verschiedene Mechanismen innerhalb der Signalisierungsverarbeitung - unabhängig von der verwendeten Multimedia-Applikation - zur Verfügung stehen. Wird eine Verbindung zwischen den Endsystemen aufgebaut, so wird zunächst die Firewall - entweder direkt oder über eine Kette von Infrastrukturkomponenten - kontaktiert. Wie die entsprechende Infrastrukturkomponente muss die Firewall nun ermitteln, welche Komponente (das Endsystem oder wiederum eine Infrastrukturkomponente) kontaktiert werden muss, um den Signalisierungspfad zwischen beiden Seiten der Firewall zu vermitteln. Im Folgenden werden hierfür drei Beispiele gegeben:

- **H.323:** Zur Firewall wird eine Q.931-Verbindung aufgebaut. Die zuerst übermittelte *Setup*-Nachricht enthält die benötigte Routing-Information. Diese Information kann eine IP-Adresse sein, eine E.164-Telefonnummer oder eine H.323-ID.
- **SIP:** An die Firewall wird eine *INVITE*-Nachricht geschickt. Diese enthält im *To*-Feld das Ziel der Kommunikation. Das Ziel kann analog einer E-Mail Adresse beschrieben werden.
- **RTSP:** Zur Firewall wird eine RTSP-Verbindung aufgebaut. Die erste übermittelte Nachricht enthält die gewünschte Adresse in Form einer URL.

Um die beiden Seiten zu verbinden, muss die Verarbeitungslogik eine Schnittstelle besitzen, über die eine Umsetzung der applikationsspezifischen Routinginformation in eine IP-Adresse erfolgen kann. Da eine Sitzung mehrere Signalisierungsströme beinhalten kann, ist es eventuell notwendig, Routinginformationen für verschiedene Ströme (eventuell auf verschiedene Art und Weise) einzuholen. Die Umsetzung der Routinginformationen kann auf verschiedene Weise erfolgen: Beispielsweise kann eine statische Konfiguration innerhalb der Firewall verwendet werden, eine DNS-Server Anfrage kann gestartet werden, eine externe Datenbank kann abgefragt werden, oder aber ein anderer Prozess innerhalb der Firewall wird angefragt, der diese Information besitzt.

Charakteristika. Eine Multimedia-Applikation kann verschiedene Signalisierungsströme innerhalb einer Sitzung verwenden. Die Verarbeitungslogik, die für die einzelnen Signalisierungsströme verwendet wird, muss in der Lage sein, mit der Verarbeitungslogik der anderen Ströme der selben Sitzung zu interagieren. Dies ist nötig, damit die Firewall die Sitzung als Einheit betrachten kann. Zusätzlich muss die Verarbeitungslogik in der Lage sein, mit der *Application Control* zu interagieren, damit Informationen (z.B. Kanalspezifikationen) mit den die Medienströme verarbeitenden Firewall-Komponenten ausgetauscht werden können.

Sicherheit. Um die notwendigen Schutzfunktionen umsetzen zu können, ist es notwendig, für die Verarbeitungslogik der einzelnen Ströme angeben zu können, wie die einzelnen auftretenden Nachrichten zu behandeln sind. Es muss für jede mögliche Nachricht angegeben werden können, ob diese weitergeleitet, verworfen oder modifiziert werden soll (entsprechend der Definition der Umsetzung in Kapitel 2 Abschnitt 2.2.1). Zusätzlich kann es erforderlich sein, zusätzliche Nachrichten auf einem der Signalisierungskanäle zu generieren, um eine Schutzfunktion umsetzen zu können. Folgende Beispiele verdeutlichen die Notwendigkeit der einzelnen Funktionen:

- In einer RTSP Nachricht kann es erwünscht sein, die Übermittlung einer *RECORD*-Nachricht zu verhindern, damit auf einem durch eine Firewall geschützten Server keine Daten abgelegt werden können.
- Es kann erwünscht sein, die in einer H.323/H.245-Nachricht übermittelten Medienbeschreibungen vor dem Weiterleiten der Nachricht zu ändern, damit bestimmte Ströme nicht über die Firewall übermittelt werden können.
- Es kann nötig sein, auf bestimmte Anforderungen (übermittelt in einer Signalisierungsnachricht) mit einer die Ablehnung der Anforderung signalisierenden Nachricht zu reagieren.

Neben den notwendigen Funktionen für die Umsetzung muss die Verarbeitungslogik für eine Nachricht entscheiden, welche der Funktionen verwendet wird und wie diese zu parameterisieren ist (Analysator und Entscheider, Kapitel 2 Abschnitt 2.2.1). Dafür ist eine Interaktion zwischen den einzelnen Verarbeitungselementen der einzelnen Signalisierungsströme einer Sitzung notwendig.

Zusammenfassung. Die gewonnenen Erkenntnisse können wiederum in Form von Designprinzipien zusammengefasst werden, die für die Umsetzung der Signalisierungsverarbeitung einer Multimedia-Firewall berücksichtigt werden sollten.

*Für die **Integration** ist eine applikationsspezifische Routingschnittstelle erforderlich. Diese muss die Kommunikation mittels verschiedener Routingmechanismen ermöglichen.*

*Für die **Verarbeitung** ist die Integration der Application Control nötig. Die Application Control sollte so ausgelegt werden, dass verschiedene Firewall-Architekturen gebildet werden können.*

*Für die Umsetzung der **Sicherheitsmechanismen** muss die Verarbeitungslogik der Signalisierung für jede Nachricht eine dedizierte Behandlung ermöglichen.*

Die hier dargestellten Designprinzipien wurden in [68], [110] und [112] beschrieben. Im Folgenden Abschnitt wird gezeigt, wie diese Designprinzipien praktisch umgesetzt werden können, und dass deren Umsetzung dazu führt, dass Firewalls in den zu unterstützenden Szenarien eingesetzt werden können.

6.4 Entwurf und Implementierung einer Signalisierungsverarbeitung

Die zuvor dargestellten Designprinzipien für die Signalisierungsverarbeitung wurden innerhalb des Systems *KOMproxyd* [113] umgesetzt. Die Architektur ist schematisch in Abbildung 39 dargestellt.

Architektur. Für jede durch das System zu unterstützende Sitzung wird innerhalb des Systems ein *Session*-Objekt instanziiert. Das *Session*-Objekt entspricht im inneren Aufbau der in Abschnitt 6.3 beschriebenen Verarbeitungslogik. Die Verarbeitungslogik für jeden Signalisierungsstrom ist in drei atomare Operationen unterteilt: *read(), parse(), write()*. Die *read()* Operation nimmt eine Nachricht entgegen, *parse()* analysiert die Nachricht und führt entsprechende Aktionen aus (z.B. Modifizieren der Nachricht, bedienen der Routing oder Application Control Schnittstellen) und *write()* sendet die Nachricht an das Ziel weiter. Jede Sitzung wird innerhalb eines *Proxy*-Objektes verwaltet (P1..Pn). Jedes *Proxy*-Objekt verwendet einen Thread. Durch diesen Thread werden die Operationen *read(), parse()* und *write()* innerhalb der *Session*-Objekte bedient. Über die Anzahl der verwendeten *Proxy*-Objekte kann das System an die Anzahl der durch das Betriebssystem zur Verfügung gestellten CPUs angepasst werden. Die *Listener*-Objekte dienen dazu, neue *Session*-Objekte bei neu zu verarbeitenden Sitzungen zu erzeugen und in die *Proxy*-Objekte einzustellen. Wird eine neue Sitzung erstellt, ist ebenfalls über das *Listener*-Objekt festgelegt für welchen Applikationstyp die Verarbeitungslogik ausgelegt wird. Die passende Verarbeitungslogik wird über dynamische Bibliotheken während der Laufzeit des Programms geladen. Das Objekt *Firewall Control* bietet die Schnittstellen, um mit der Firewall-Seite zu kommunizieren. Wird beispielsweise innerhalb eines *Session*-Objektes in der Funktion *parse()* festgestellt, dass ein Medienstrom ausgehandelt wurde, so wird über die Funktion *fw_add_fil()* innerhalb des *Firewall Control*-Objektes diese Information an die Firewall weitergeleitet. Bis eine Bestätigung eintrifft, dass diese Information auf Firewall-Seite verwendet werden konnte, wird die Sitzung schlafen gelegt. Je nach verwendeter Firewall können verschiedene *Firewall Control*-Objekte innerhalb des Systems verwendet werden, diese

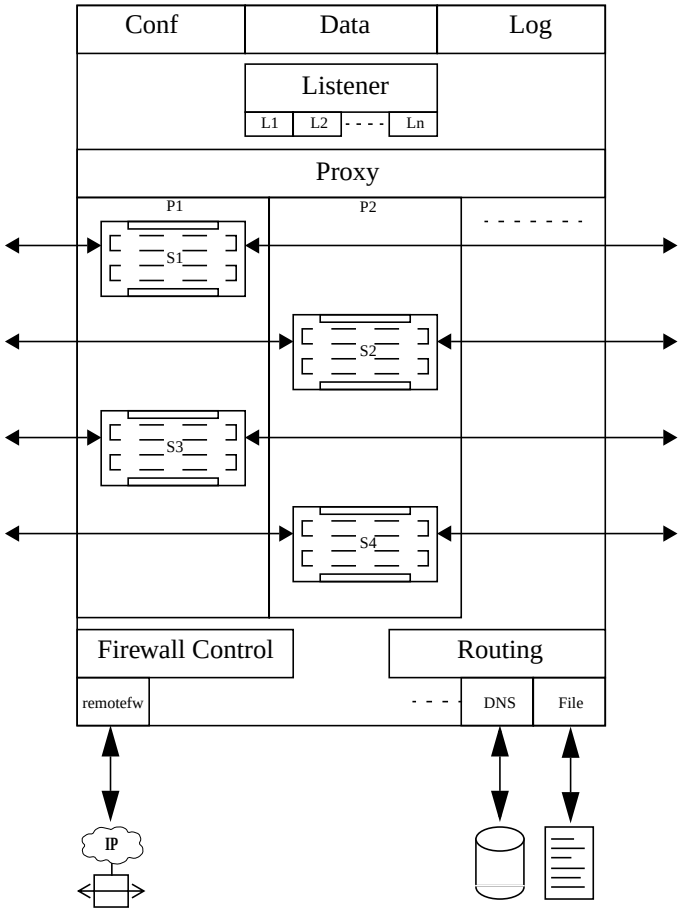


Abbildung 39: *KOMproxyd*-Systemarchitektur

werden als dynamische Bibliothek bei Systemstart geladen. Die Routing-Schnittstelle besitzt generische Funktionen (z.B. Zielbestimmung über ein Konfigurationsfile), ist aber durch Funktionen, die innerhalb der *Session*-Objekte definiert werden können, erweiterbar, um ein applikationsspezifisches Routing ermöglichen zu können. Das *Conf*-Objekt bietet die Schnittstelle zum Auslesen einer Konfigurationsdatei, das *Log*-Objekt die Schnittstelle für ein Audit. *Data* (realisiert innerhalb des *Conf*-Objektes) bietet die Möglichkeit, Informationen die systemweit benötigt werden, abzulegen (z.B. Informationen die zwischen *Session*-Objekten ausgetauscht werden sollen).

Applikationsmodule. Wie beschrieben, wird die Verarbeitungslogik innerhalb der *Session*-Objekte durch das Laden eines Moduls (ein spezifisches *Parser*-Objekt innerhalb des *Session*-Objektes) für einen Applikationstyp angepasst. Soll ein neuer Applikationstyp durch die Firewall unterstützt werden, muss ein neues Modul für diese Applikation entworfen werden. Im Folgenden werden die bereits implementierten Module kurz beschrieben:

- **ras:** Das *ras*-Modul wird verwendet, um die RAS-Kommunikation zwischen benachbarten Gatekeepern zu analysieren. Das Modul legt die durch die Analyse der RAS-Kommunikation gewonnenen Routinginformationen innerhalb der Datenbank des *Conf*-Objektes ab. Diese Informationen stehen dann den H.323-*Session*-Objekten über die Routingschnittstelle zur Verfügung. Für jede RAS-Kommunikation wird nur ein RAS-Session-Objekt verwendet, da die UDP-basierte RAS Kommunikation nicht über verschiedene Ströme diskriminiert werden kann. Dieses wird nicht durch einen Listener gestartet, sondern direkt bei Systemstart angelegt. Zur Umsetzung der Sicherheitsanforderungen kann innerhalb der Konfiguration des Systems festgelegt werden, welche RAS Nachrichten über die Firewall weitergeleitet werden sollen. Für die Realisierung dieses Moduls wurde der OpenH323-Protokollstack [69] verwendet.
- **h323:** Das *h323*-Modul beinhaltet die zur Verarbeitung einer H.323-Sitzung notwendige Verarbeitungslogik. Durch diese Modul werden Q.931- und H.245-Ströme einer Sitzung verarbeitet. Ausgehandelte Medienströme werden über die Firewall Control Schnittstelle an die Firewall übermittelt. Zur Umsetzung der Sicherheitsanforderungen kann über das Konfigurationsfile festgelegt werden, welche Nachrichtentypen über die Firewall weitergeleitet werden sollen. Es ist ebenfalls möglich, das Aushandeln spezieller Medienströme zu unterbinden (z.B. T.120 zur Datenübertragung). Für die Realisierung dieses Moduls wurde ebenfalls der OpenH323 Protokollstack verwendet.
- **sip:** Das *sip*-Modul beinhaltet die Verarbeitungslogik für eine SIP-Sitzung. Da die SIP-Signalisierung über UDP läuft, wird ein SIP-*Session*-Objekt für die Behandlung mehrerer SIP-Sessions verwendet. Die einzelnen Sessions werden über die SIP-CallID innerhalb des Moduls diskriminiert. Zur Umsetzung der Sicherheitsanforderungen kann die Übermittlung bestimmter Nachrichtentypen unterbunden werden. Dieses Modul baut auf der OSIP-Bibliothek [114] auf.

- **rtsp:** Das *rtsp*-Modul beinhaltet die Verarbeitungslogik für eine RTSP-Sitzung. Zur Umsetzung der Sicherheitsanforderungen kann die Übermittlung bestimmter Nachrichtentypen unterbunden werden. Dieses Modul baut auf dem KOM-Player [115] auf.
- **test:** Das *test*-Modul enthält die Verarbeitungslogik für das in Kapitel 7 beschriebene Testprotokoll. Dieser Applikationstyp wird für die Bestimmung der Leistungskenngrößen einer Firewall verwendet.
- **Andere:** Es existieren ebenfalls Module für weitere Applikationen (generic, ftp, fw).

Jedes *Session*-Objekt besitzt zusätzlich die Fähigkeit, auch Medienströme zu verarbeiten. Durch eine entsprechenden Konfigurationseinstellung ist die Verarbeitungslogik der Sitzung in der Lage, nicht nur Signalisierungsströme, sondern auch Medienströme zu verarbeiten. Dies ist entsprechend dem gegebenen Designprinzip in Kapitel 3 zwar kein geeigneter Betriebsmodus, besitzt aber einige Vorteile in der Praxis. Dadurch ist es z.B. möglich das System als Proxy (bezeichnet als Proxy-Modus) zu betreiben, wenn keine geeigneten Firewall-Komponenten zur Ansteuerung über die *Firewall Control* zur Verfügung stehen. Bis zu einer geringen Anzahl parallel zu unterstützenden Sitzungen (siehe Kapitel 7) kann das System in dieser Konfiguration verwendet werden.

Firewall-Module. Die Firewall Control Schnittstelle wird ebenfalls über das dynamische Einbinden eines Firewall-Moduls konfiguriert. Die folgenden Module wurden implementiert.

- **ipflfw:** Dieses Modul ermöglicht es *KOMproxyd*, über entsprechende *ioctl()* Systemaufrufe mit einem *ip-filter* [104] Paketfilter zu interagieren. Wird dieses Modul verwendet, befinden sich *Application Control* und *Firewall Control* auf dem selben System. Die resultierende Firewall entspricht in diesem Fall einem Hybridsystem.
- **remotefw:** Dieses Modul wird verwendet, um mit einer *Application Control* zu interagieren, die über ein IP-Netzwerk zu erreichen ist. Alle Systemaufrufe werden über ein simples UDP-basiertes FCP transportiert. Die Firewall-Seite kann umgesetzt werden, indem dort eine weitere Instanz des *KOMproxyd* verwendet wird. Als Sitzungsmodul wird dort das *fw*-Modul verwendet, als Firewall-Modul beispielsweise das *ipflfw*-Modul. Der Parser des *fw*-Moduls extrahiert die übermittelten Informationen und übergibt sie an die lokale *fw*-Schnittstelle. Die folgende Abbildung zeigt diese Konfiguration:

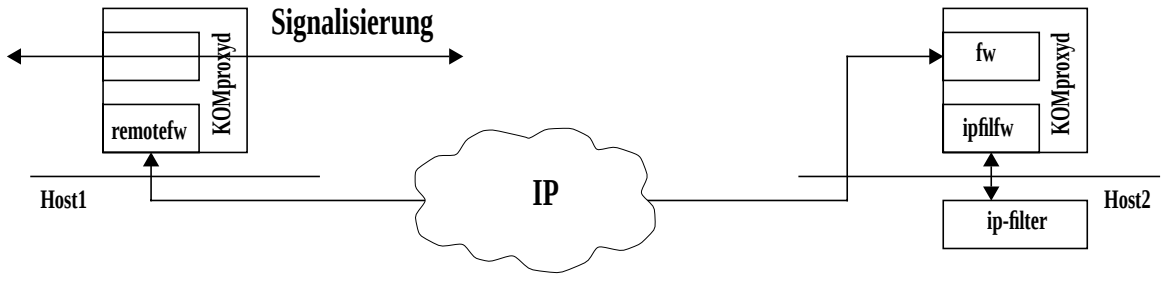


Abbildung 40: *KOMproxyd remote*fw

- **testfw:** Dieses Modul wird zu Testzwecken verwendet. In diesem Fall werden alle übermittelten Informationen (z.B. eine Filterspezifikation) nur bestätigt ohne dass tatsächlich Aktionen ausgeführt werden.

Zusammenfassung. Das im Rahmen dieser Arbeit realisierte System *KOMproxyd* unterstützt die in Abschnitt 6.2 und Abschnitt 6.3 aufgestellten Designprinzipien. Es kann mit diesem System ebenfalls das in Kapitel 3 beschriebene Designprinzip der Trennung der Signalisierungs- und Medienströme umgesetzt werden. Im Folgenden werden verschiedene Beispiele aufgeführt, in denen das System eingesetzt wird. Es wird gezeigt, dass die Umsetzung der aufgestellten Designprinzipien es ermöglichen problematische Applikations-Szenarien zu unterstützen. In Kapitel 7 wird auf die Leistungsfähigkeit des Systems genauer eingegangen.

6.5 Praktischer Einsatz der Implementierung

Ein Beispiel für den Einsatz des *KOMproxyd* in der Praxis ist der Videokonferenzdienst des Deutschen Forschungsnetzes (DFN), der sich zur Zeit noch im Aufbau befindet. Der vom DFN angebotene Videokonferenzdienst basiert auf dem H.323-Protokoll und wird verwendet, um Wissenschaftlern an verschiedenen Standorten die Möglichkeit zu geben, sich über Konferenzschaltungen auszutauschen. Im folgenden Abschnitt wird das im DFN verwendete H.323-Szenario beschrieben.

Szenario. Ziel des DFN ist es, einen auf dem H.323-Protokoll basierenden Videokonferenzdienst anzubieten. Dazu werden vom DFN die notwendigen MCU-Einheiten sowie eine für das Call-Routing notwendige Gatekeeper-Struktur bereitgestellt. Die bereitgestellte Gatekeeper-Struktur ist in Abbildung 41 dargestellt.

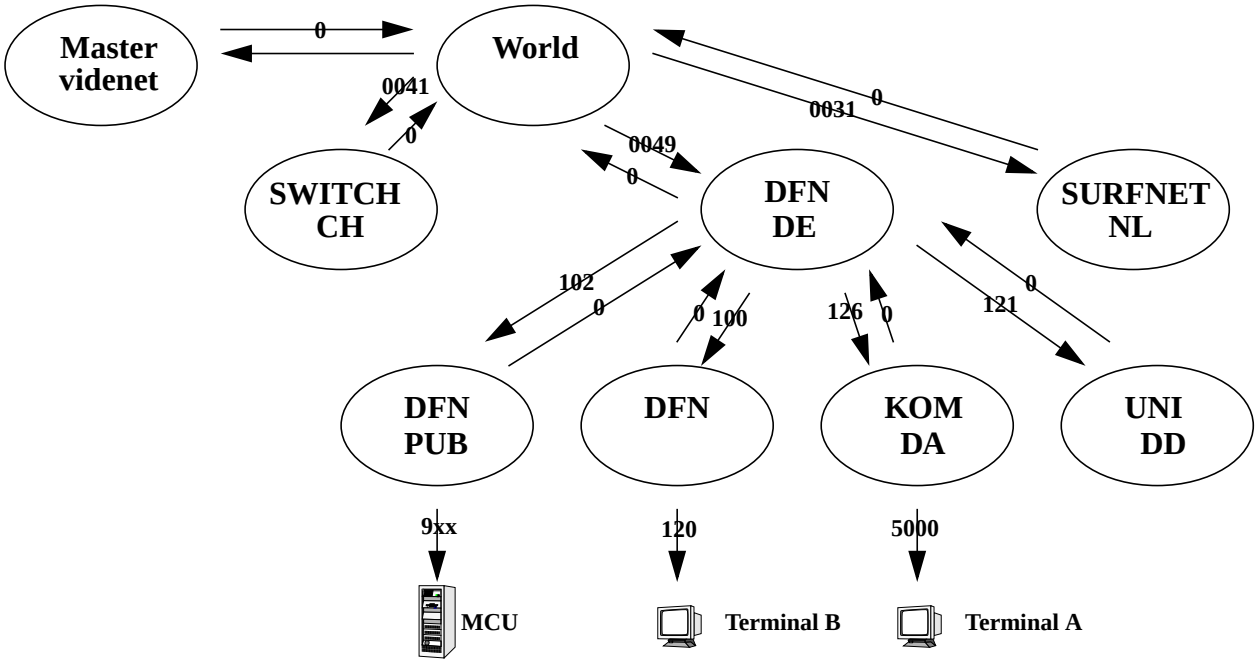


Abbildung 41: Internationale Gatekeeper-Struktur

Der DFN stellt einen Gatekeeper bereit (DFN DE), der zwischen den einzelnen H.323-Zonen innerhalb Deutschlands (z.B. KOM DA, UNI DD,...) für das Call Routing verwendet werden kann und eine internationale Anbindung bereitstellt. Zusätzlich wird vom DFN ein Nummernplan bereitgestellt, der es ermöglicht neue Zonen in das Gesamtsystem zu integrieren. Der DFN betreibt für den operator-unterstützten Dienst eine eigene H.323-Zone (DFN) sowie eine öffentliche Zone (DFN PUB), in der die für den Videokonferenzdienst notwendigen MCUs angeordnet sind. Das folgende Beispiel stellt dar, wie ein Ruf zwischen zwei Zonen innerhalb Deutschlands in diesem System vermittelt wird:

- In diesem Beispiel wird angenommen, dass das Terminal (Terminal A) mit der Nummer 5000 innerhalb der Zone KOM-DA (das Institut KOM an der Technischen Univer-

sität Darmstadt) das Terminal (Terminal B) mit der Rufnummer 120 in der Zone DFN (DFN Geschäftsstelle in Berlin) anruft.

- Das rufende Terminal A ist an dem Gatekeeper der Zone KOM-DA angemeldet, Terminal B ist an dem Gatekeeper der Zone DFN angemeldet.
- Um Terminal B zu kontaktieren, wählt Terminal A die Rufnummer 0100120.
- Der Gatekeeper der Zone KOM-DA erkennt, dass das Ziel-Terminal mit der Rufnummer 0100120 nicht bei ihm selbst registriert ist. Anhand der *Neighbour Table* des Gatekeepers muss nun entschieden werden, wie das Ziel gefunden wird. Der Gatekeeper besitzt einen *Neighbour Entry*, der für das Präfix 0 auf den DFN-DE Gatekeeper verweist. Der Gatekeeper der Zone KOM-DA sendet nun einen entsprechenden *Location Request (LRQ)* für die Rufnummer 100120[1] an den DFN-DE Gatekeeper.
- Der DFN-DE Gatekeeper, der *Neighbour Entries* für alle innerhalb Deutschlands verwendeten H.323-Zonen besitzt, leitet die *LRQ*-Anfrage entsprechend seines *Neighbour Entry* für die Zone mit dem Prefix 100 an den Gatekeeper der Zone DFN weiter.
- Der Gatekeeper der Zone DFN bestätigt nun die Anfrage mit einem *Location Confirm (LCF)*, da ein Terminal mit der Rufnummer 120 (bzw. 100120) bei ihm registriert ist. Die *LCF*-Nachricht wird über den DFN-DE Gatekeeper an den KOM-DA Gatekeeper weitergeleitet. Die *LCF*-Nachricht enthält die *Q.931*-Adresse von Terminal B. Je nach Gatekeeper Policy (z.b. gatekeeper routed call), wird diese Adresse von den Gatekeepern vor dem Weiterleiten eventuell durch die eigene ersetzt.
- Schließlich kann die H.323-Verbindung zwischen den Terminals aufgebaut werden.

Soll eine MCU verwendet werden, um beispielsweise eine Konferenz mit mehreren Teilnehmern durchzuführen, verbinden sich alle Konferenzteilnehmer mit der MCU. Zuvor muss über geeignete Mechanismen gesichert werden, dass die Ressourcen innerhalb der MCU zum gewünschten Zeitpunkt für die Dauer der Konferenz bereitgestellt werden können. Die einzelnen Konferenzteilnehmer bauen zwischen ihrem Terminal und der MCU jeweils eine Standard-H.323-Verbindung auf, die MCU übernimmt das Mischen der Audio- und Videodaten. Die einzelnen zur MCU aufgebauten Verbindungen der Konferenzteilnehmer unterscheiden sich dabei nicht von einer Verbindung zu einem normalen Terminal. Eine in dieser Architektur verwendete Firewall, die die Kommunikation zwischen zwei Terminals unterstützt, unterstützt damit auch automatisch die Kommunikation mit einer MCU.

Integration und Verarbeitung. Eine H.323-Firewall-Lösung innerhalb des DFN muss sich in die im vorigen Abschnitt beschriebene H.323-Struktur einfügen. Um festlegen zu können, wie in die in Abbildung 41 dargestellte H.323-Struktur eine Firewall einfügt werden kann, müssen einige Randbedingungen betrachtet werden. Eine H.323-Zone entspricht in der Regel dem durch eine Firewall geschützen Bereich. Eine Institution wird eine eigene H.323-Zone innerhalb des

1. Gatekeeper können den Prefix vor Weiterleitung der Anfrage von der Rufnummer entfernen. Dies hängt von der Konfiguration des Gatekeepers sowie der Gatekeeper-Implementierung ab.

eigenen Netzes verwenden, um dort den verwendeten Nummernraum bzw. die möglichen H.323-Dienste eigenständig verwalten und konfigurieren zu können. Dieselbe Institution wird auch eine Firewall verwenden, um das interne Netz gegen Angriffe von außerhalb zu schützen.

Aus den oben genannten Randbedingungen ergibt sich, dass eine H.323-Firewall innerhalb des

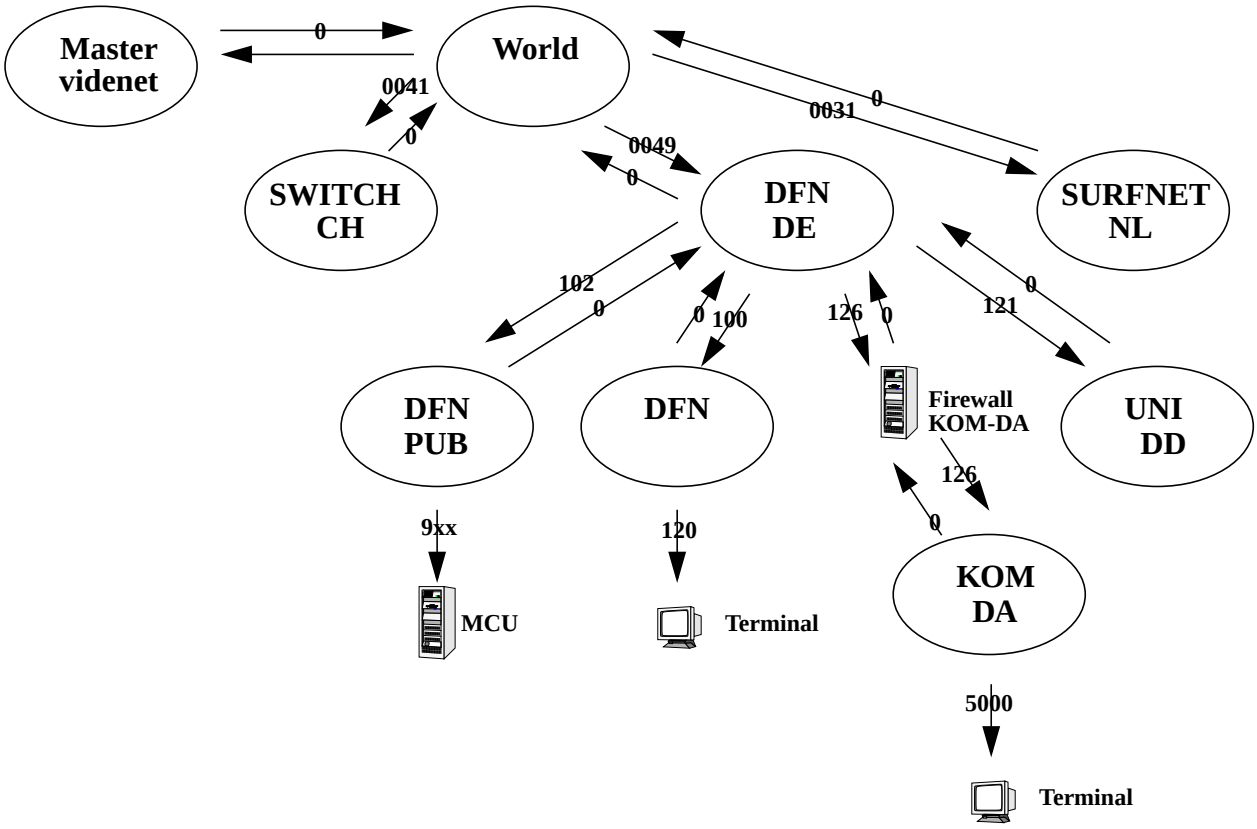


Abbildung 42: Internationale Gatekeeper-Struktur mit Firewall

DFN-Szenarios an den Übergängen zwischen verschiedenen H.323-Zonen angeordnet werden sollte. Abbildung 42 zeigt die DFN-H.323-Struktur mit einer Firewall, die das interne Netzwerk (und damit auch die H.323 Zone KOM-DA) des Institutes KOM in Darmstadt absichert. Die so integrierte Firewall muss in der Lage sein, sich entsprechend den gegebenen Designprinzipien in die Gatekeeper Hierarchie einzufügen. Es ergibt sich nun folgende, veränderte Rufvermittlung:

- In diesem Beispiel wird ebenfalls angenommen, dass das Terminal (Terminal A) mit der Nummer 5000 innerhalb der Zone KOM-DA das Terminal (Terminal B) mit der Rufnummer 120 in der Zone DFN anruft.
- Das rufende Terminal A ist bei dem Gatekeeper der Zone KOM-DA angemeldet, Terminal B ist bei dem Gatekeeper der Zone DFN angemeldet.
- Um Terminal B zu kontaktieren, wählt Terminal A die Rufnummer 0100120.
- Der Gatekeeper der Zone KOM-DA erkennt, dass das Ziel-Terminal mit der Rufnummer 0100120 nicht bei ihm selbst registriert ist. Der Gatekeeper besitzt einen *Neigh-*

bour Entry, der für den Prefix 0 auf die Firewall der Zone KOM-DA verweist. Der Gatekeeper der Zone KOM-DA sendet nun einen entsprechenden *Location Request (LRQ)* für die Rufnummer 0100120 an die KOM-DA Firewall.

- Die Firewall erkennt, dass das Ziel-Terminal mit der Rufnummer 0100120 über den Gatekeeper DFN-DE zu erreichen ist (*Neighbour Table*). Die Firewall der Zone KOM-DA sendet nun einen entsprechenden *Location Request (LRQ)* für die Rufnummer 100120 an den DFN-DE Gatekeeper.
- Der DFN-DE Gatekeeper, der *Neighbour Entry* für alle innerhalb Deutschlands verwendeten H.323-Zonen besitzt, leitet die *LRQ*-Anfrage entsprechend seines *Neighbour Entry* für die Zone mit dem Präfix 100 an den Gatekeeper der Zone DFN weiter.
- Der Gatekeeper der Zone DFN bestätigt nun die Anfrage mit einem *Location Confirm (LCF)*, da ein Terminal mit der Rufnummer 120 (bzw. 100120) bei ihm registriert ist. Die *LCF*-Nachricht wird über den DFN-DE Gatekeeper und über die KOM-DA Firewall an den KOM-DA Gatekeeper weitergeleitet. Die *LCF*-Nachricht enthält die *Q.931*-Adresse von Terminal B. Je nach Gatekeeper Policy (z.b. gatekeeper routed call) wird diese Adresse von den Gatekeepern und der Firewall vor dem Weiterleiten eventuell durch die eigene ersetzt.
- Schließlich kann die H.323-Verbindung zwischen den Terminals über die Firewall aufgebaut werden.

Es wurde damit gezeigt, dass *KOMproxyd* innerhalb des DFN-Szenarios effizient integriert werden kann. Durch das beschriebene Verfahren ist es möglich die auftretenden Szenarien zu unterstützen, so z.B. auch Szenarien, in denen Firewalls mit NAT verwendet werden. Da *KOMproxyd* die Möglichkeit besitzt, Sicherheitsfunktionen umzusetzen - z.B. das Weiterleiten bestimmter PDUs - können durch Einsatz dieses Systems auch die meisten der in Kapitel 3 beschriebene Angriffe verhindert werden (z.B. Angriffe auf den durch die Firewall geschützten Gatekeeper).

Zusammenfassung. Anhand des oben beschriebenen Beispiels wurde gezeigt, das die aufgestellten Designprinzipien es ermöglichen, eine Firewall in komplexen Szenarien zu verwenden. Das in dieser Arbeit vorgestellte System *KOMproxyd* wird innerhalb des beschriebenen DFN-Szenarios produktiv verwendet. *KOMproxyd* wird ebenfalls durch die niederländische Organisation GigaPort (ähnlich dem DFN) verwendet. In einer dort durchgeführten Studie [116] wurde sich für die Verwendung des *KOMproxyd* System innerhalb der dortigen H.323-Infrastruktur entschieden. In weiteren Versuchen (siehe auch [57]) wurde ebenfalls gezeigt, das *KOMproxyd* auch in einem RTSP-Szenario und SIP-Szenario effizient verwendet werden kann.

6.6 Zusammenfassung

In diesem Kapitel wurde untersucht, wie eine Signalisierungsverarbeitung innerhalb einer Multimedia-Firewall umgesetzt werden muss. Es wurden dafür die zwei wesentlichen Aspekte “Integration” und “Verarbeitung” untersucht. Zum einen muss die Signalisierungsverarbeitung einer Firewall in das verwendete Kommunikationsszenario der Multimedia-Applikationen eingebunden werden (Integration). Zum anderen muss die Signalisierungsverarbeitung so ausgelegt werden, dass mit den in Kapitel 3 beschriebenen Charakteristika von Multimedia-Applikationen umgegangen werden kann (Verarbeitung). Um beiden Aspekten gerecht werden zu können, wurden Designprinzipien formuliert, die eine effiziente Umsetzung der Signalisierungsverarbeitung ermöglichen.

Anhand der im Rahmen dieser Arbeit durchgeführten prototypischen Implementierung *KOMproxyd* wurde gezeigt, dass die Verwendung der aufgestellten Designprinzipien tatsächlich die aufgezeigten Probleme löst. Darüber hinaus kann die Software als Baukasten verwendet werden, um verschiedene Firewall-Architekturen für Multimedia-Applikationen zu realisieren. Der im Rahmen dieser Arbeit entwickelte Prototyp *KOMproxyd* ist unter [113] verfügbar. Dieses System wird zur Zeit von mehreren Institutionen (z.B. DFN, SurfNet, GigaPort) innerhalb von H.323-basierten Szenarien eingesetzt.

Kapitel 7: Leistungsbewertung von Firewall-Architekturen

In diesem Kapitel wird die Leistungsfähigkeit von Firewall-Architekturen untersucht. Es werden die in den vorherigen Kapiteln beschriebenen und diskutierten Firewall-Architekturen hinsichtlich ihrer Leistungsfähigkeit untersucht und bewertet.

7.1 Motivation und Zielsetzung

In Kapitel 3 wurde das Designprinzip *Trennung von Signalisierungs- und Medienpfad* aufgestellt. In den vorangegangenen Kapiteln wurde gezeigt, welche Architekturvorteile sich ergeben, wenn dieses Designprinzip verwendet wird. Es wurde ebenfalls betrachtet, wie einzelne Bausteine zur Realisierung einer Firewall, die dieses Designprinzip verwendet, umgesetzt werden können bzw. müssen.

Motivation. Neben den in den vorangegangenen Kapiteln diskutierten Kriterien ist die Leistungsfähigkeit ein weiteres wichtiges Kriterium zur Beurteilung einer Firewall. Eine Firewall, die den einzigen Übergangspunkt zwischen zwei Netzen darstellt, was zur Erbringung ihrer Schutzfunktion nötig ist, läuft grundsätzlich Gefahr, als Flaschenhals des Systems zu wirken. Es ist deshalb wichtig, Firewall-Architekturen, die eine große Leistungsfähigkeit besitzen, zu entwickeln bzw. zu verwenden. Die Leistungsfähigkeit einer Firewall spielt insbesondere dann eine Rolle, wenn man als Einsatzort nicht den Übergang eines Firmennetzes zum Internet, sondern beispielsweise den Übergang zwischen den Netzen zweier Netzanbieter betrachtet.

Randbedingungen. In diesem Kapitel liegt der Fokus auf der Klasse der Telefonie-Applikationen. Die gewonnenen Erkenntnisse können aber in der Regel auf andere Klassen von Multimedia-Applikationen (z.B. VoD-Applikationen) übertragen werden.

Zielsetzung. Ziel dieses Kapitels ist es, zu untersuchen, welche Auswirkung die Umsetzung des Designprinzips *Trennung von Signalisierungs und Medienpfad* auf die Leistungsfähigkeit einer Firewall bzw. der ihr zugrunde liegenden Firewall-Architektur hat. In diesem Kapitel wird daher zunächst untersucht, welche Kenngrößen eine Bewertung der Leistungsfähigkeit einer Multimedia-Firewall ermöglichen. Anschließend wird untersucht, wie diese Kenngrößen für einzelne Firewall-Architekturen ermittelt werden können. Abschließend werden die Kenngrößen für verschiedene Architekturen bestimmt und eine Bewertung der verschiedenen Architekturen durchgeführt.

7.2 Dienstgüteparameter

Damit Aussagen über die Leistungsfähigkeit einer Firewall für Multimedia-Applikationen getroffen werden können, müssen geeignete Kenngrößen, die eine objektive Bewertung dieser Aussagen zulassen, definiert werden. Im Folgenden wird zunächst betrachtet, welche Kenngrößen zur Beschreibung der Qualität von Sprach- und Videoverbindungen verwendet werden können bzw. müssen. Aus diesen werden in Abschnitt 7.3 die Kenngrößen zur Bewertung der Leistungsfähigkeit einer Firewall abgeleitet. Um die Qualität von Sprach- und Videoverbindungen zu beschreiben, müssen sowohl die Signalisierungsebene als auch die Transportebene betrachtet werden. Beide Ebenen spielen bezüglich subjektiv empfundener und objektiv gemessener Qualität eine entscheidende Rolle.

7.2.1 Signalisierung

Die Güte der Signalisierungsebene wirkt sich insbesondere auf den Aufbau einer Sitzung aus. Ist die für den Aufbau nötige Zeit zu lang, kann beispielsweise der Benutzer einer Multimedia-Applikation dies als störend bzw. sogar inakzeptabel empfinden.

Sitzungsaufbaudauer. Die Sitzungsaufbaudauer einer Multimedia-Applikation wird innerhalb der vorliegenden Arbeit folgendermaßen definiert:

> *Die **Sitzungsaufbaudauer (Session Setup Time)** T_S ist die Zeit, die vom Beginn des Aufbaus des Signalisierungskanals bis zum Beginn des Medientransports benötigt wird.*

Grenzwert und exakte Definition der Sitzungsaufbaudauer hängt vom Typ der untersuchten Applikation ab. Bei einer VoD-Applikation wird es von einem Benutzer in der Regel nicht als sehr störend empfunden, wenn nach Anforderung einige Sekunden vergehen, bis mit dem Abspielen des gewünschten Films begonnen wird [26]. Bei einer Telefonie-Applikation hingegen wird es von einem Benutzer als sehr störend empfunden, wenn nach dem Annehmen des Gesprächs einige Sekunden vergehen, bevor gesprochen werden kann [118].

Die Sitzungsaufbaudauer einer Multimedia-Applikation kann, abhängig von dem jeweilig betrachteten Applikationstyp, in mehrere Schritte unterteilt werden, wobei die einzelnen Teilschritte wiederum verschiedenen Anforderungen unterliegen.

Dieses wird im Folgenden für die Klasse der Telefonie-Applikationen näher betrachtet. In [119] werden für eine H.323-Applikation die in Abbildung 43 dargestellten Teilschritte innerhalb der Sitzungsaufbaudauer angegeben:

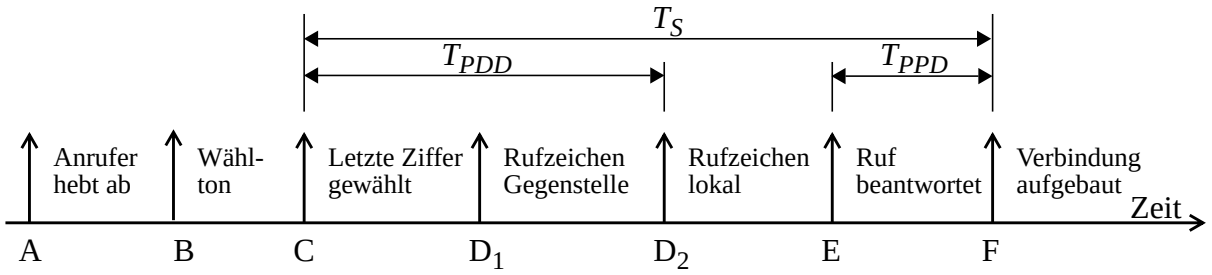


Abbildung 43: Zeitabläufe beim Sitzungsaufbau einer H.323-Applikation

Die nach oben stehender Definition gegebene Sitzungsaufbaudauer ergibt sich dann zu $T_S = F - C$. Zusätzlich werden der Rufverzug (Post Dial Delay) $T_{PDD} = D_2 - C$, sowie die Meldedauer (Post Pickup Delay) $T_{PPD} = F - E$ definiert. Der Rufverzug gibt die Zeit zwischen dem Wählen der letzten Ziffer und dem Klingeln auf lokaler Seite an; die Meldedauer gibt die Zeit zwischen dem Annehmen des Gespräches und der Herstellung der eigentlichen Verbindung an. Insbesondere die Meldedauer ist ein kritischer Wert. Ist diese Verzögerung zu groß, werden die ersten Worte des Angerufenen verschluckt, da die Medienströme noch nicht geschaltet sind. Die Meldedauer sollte daher möglichst kleiner sein als die Zeit, die ein Mensch braucht, um den Telefonhörer seines Telefons abzunehmen und ans Ohr zu führen.

Während die einzelnen, die Güte beeinflussenden Parameter in der Literatur beschrieben werden, sind Richtlinien oder Spezifikationen, die die Grenzwerte der Parameter beschreiben, Mangelware. In [118] werden zulässige Maximalwerte für Rufverzug und Meldedauer für das ISDN-Netz angegeben. Diese können, wie auch in [119] durchgeführt, auf IP-Netze übertragen werden. Je nach Szenario - bestimmt durch Parameter wie z.B. Netzgröße, Netzauslastung oder Distanz - ergeben sich dann Grenzwerte, die für den Rufverzug im Bereich von 3 bis 8 Sekunden und für die Meldedauer im Bereich von 0.75 bis 2 Sekunden liegen. Eine weiterführende Beschreibung des Parameters Sitzungsaufbaudauer findet sich in [120].

7.2.2 Medienströme

Wenn die Medienströme nicht einem Mindestanspruch genügen, ist die eigentliche Kommunikation erschwert bzw. unmöglich. Vorstellbar sind Auswirkungen wie starkes Echo, lange Verzögerungszeiten oder Rauschen. Im Folgenden werden die wesentlichen Parameter zur Beschreibung der Güte der Medienströme definiert.

Verzögerung. Für die Verzögerung gibt es in der Literatur verschiedene Definitionen. Wird beispielsweise ein Audiosignal von einer Quelle zu einer Senke transportiert, ist nicht genau festgelegt, welche Teile der Übertragungsstrecke zur Bestimmung der Verzögerung verwendet werden müssen. Das Audiosignal wird zunächst digitalisiert und mit Hilfe eines zuvor festgelegten Verfahrens kodiert. Danach wird es über das zwischen Quelle und Senke gelegene Netz transportiert, durchläuft einen Puffer zum Ausgleich des Jitters und wird schließlich dekodiert und nach einer D/A-Wandlung über den Lautsprecher ausgegeben. Jeder dieser auszuführenden Schritte benötigt eine gewisse Zeit, die der Gesamtverzögerung (Ende-zu-Ende Verzögerung) zugerechnet werden muss. In dieser Arbeit wird als Verzögerung nur die durch den Transport der Daten über das Netzwerk verursachte Verzögerung verstanden. Somit kann für jedes gesendete Paket eines Medienstroms die Verzögerung bestimmt werden.

> *Die **Verzögerung (Delay)** $T_{D(i)}$ ist die Zeit, die benötigt wird, um ein Paket i eines Medienstroms über das zwischen Quelle und Senke liegende Netzwerk zu transportieren.*

Für verschiedene Applikationstypen sowie Szenarien, in denen diese verwendet werden, können Grenzwerte für die Verzögerung angegeben werden, die bei Einhaltung zu einer für den Benutzer akzeptablen Qualität der Applikation führen. Nimmt man wiederum eine H.323-Applikation als

Beispiel, ergibt sich entsprechend [119] eine maximale Obergrenze von 50 ms bis 100 ms für die Verzögerung. Auf die Bestimmung von Verzögerungsgrenzwerten wird in [26] detailliert eingegangen.

Jitter. Die Varianz der Verzögerung der einzelnen Pakete eines Medienstroms wird als Jitter bezeichnet. Die Varianz zweier aufeinander folgender Pakete eines Medienstroms wird wie folgt definiert:

*Der **Jitter** $T_{J(i)}$ zweier aufeinander folgender Pakete eines Datenstroms ist gegeben durch* $T_{J(i)} = \left| T_{D(i)} - T_{D(i+1)} \right|$.

Grenzwerte für den Jitter können wiederum nur anwendungsspezifisch gegeben werden. In [119] wird für eine H.323-Applikation ein Bereich von 10 ms bis 40 ms als Obergrenze angegeben.

Paketverlust. Der auftretende Paketverlust wird ebenfalls in der Literatur verschieden angegeben. Einerseits zählen die beim Transport der Pakete eines Medienstroms auftretenden Paketverluste, andererseits können aber auch zu spät eintreffende Medienpakete als Paketverlust gezählt werden, da diese beim Empfänger nicht mehr verwendet werden können. In dieser Arbeit werden nur die tatsächlich beim Transport verlorengegangenen Pakete als Paketverlust verstanden.

*Als **Paketverlust (Loss)** L wird der prozentuale Anteil an verlorengegangenen Paketen beim Transport über das zwischen Quelle und Senke liegende Netz bezeichnet.*

Grenzwerte für Paketverluste können nicht generell angegeben werden. Verschiedene Medienkodierungsverfahren können unterschiedlich gut mit Paketverlusten umgehen. Generell sind Paketverluste jedoch soweit wie möglich zu minimieren. In [119] wird für eine H.323-Applikation ein Bereich von 0.5% bis 2% als Obergrenze angegeben.

7.3 Leistungskenngrößen von Firewall-Architekturen

Soll die Leistungsfähigkeit einer Firewall bewertet oder verbessert werden, so müssen zunächst die dem Begriff *Leistung* zugrunde liegenden Kenngrößen identifiziert werden. Für eine Multimedia-Firewall sind dies zum einen die aus den in Abschnitt 7.2 beschriebenen Dienstgüteparametern abgeleiteten Kenngrößen sowie zum anderen die maximale Anzahl an Multimedia-Sitzungen, die gleichzeitig von einer Firewall unterstützt werden können.

Im Folgenden werden die einzelnen Leistungskenngrößen dargestellt und beschrieben. Es wird ebenfalls untersucht, welche Faktoren Einfluss auf die jeweilige Kenngröße haben. Dadurch können Faktoren identifiziert werden, deren Beeinflussung zu einer Optimierung der Leistungsfähigkeit einer Firewall führen.

7.3.1 Aus Dienstgüteparametern abgeleitete Leistungskenngrößen

Die festzulegenden Leistungskenngrößen sollen es ermöglichen, eine objektive und quantifizierbare Aussage darüber zu treffen, ob die Leistungsfähigkeit einer Firewall-Architektur besser ist als die einer anderen. Zur Festlegung der Kenngrößen wurden die in Abschnitt 7.2 beschriebenen Dienstgüteparameter verwendet, die die Güte hinsichtlich der Leistung einer Multimedia-Applikation beschreiben. Es wird folgende Definition verwendet:

> *Eine Leistungskenngröße G gibt an, wieviel Prozent der gegebenen Obergrenze eines Dienstgüteparameters für eine spezifische Multimedia-Applikation durch das Einbringen der Firewall in den Kommunikationsweg durch die Firewall verwendet werden.*

Die einzelnen Kenngrößen hängen ab von der Anzahl n gleichzeitig über die Firewall abgewikkelter Sitzungen eines spezifischen Applikationstyps. Die auf der Sitzungsaufbaudauer basierende Kenngröße G_{T_S} wird über die durch das Einbringen der Firewall erzeugte zusätzliche Sitzungsaufbaudauer ΔT_S sowie der für die betrachtete Applikation maximal zulässigen Sitzungsaufbaudauer $T_{S_{max}}$ bestimmt.

$$G_{T_S}(n) = \frac{\Delta T_S(n)}{T_{S_{max}}} \tag{2}$$

Analog können die restlichen Kenngrößen wie folgt bestimmt werden:

$$G_{T_D}(n) = \frac{\Delta T_D(n)}{T_{D_{max}}} \tag{3}$$

$$G_{T_J}(n) = \frac{\Delta T_J(n)}{T_{J_{max}}} \tag{4}$$

$$G_L(n) = \frac{\Delta L(n)}{L_{max}} \tag{5}$$

Diese Kenngrößen können zusammen mit der nachfolgend beschriebenen Gesamtleistung für verschiedene Firewalls bestimmt werden. Dies ermöglicht dann einen Vergleich der verschiedenen Systeme. Eine allgemeingültige Aussage über die vorzunehmende Gewichtung der einzelnen Kenngrößen für eine Bewertung kann allerdings nicht gegeben werden.

7.3.2 Gesamtleistung

Die Gesamtleistung einer Multimedia-Firewall wird in der vorliegenden Arbeit folgendermaßen definiert:

> *Die Gesamtleistung N einer Multimedia-Firewall ist gegeben durch die Anzahl an gleichartigen Multimedia-Sitzungen, die gleichzeitig ihre Kommunikation über die Firewall abwikkeln können.*

Nach dieser Definition kann die Gesamtleistung immer nur für einen Applikationstyp (z.B. H.323, RTSP) angegeben werden. Für verschiedene Applikationstypen kann eine Firewall eine unterschiedliche Gesamtleistung erreichen.

Die Gesamtleistung einer Multimedia-Firewall wird im Wesentlichen durch die Faktoren “Regelinstantiierung”, “Durchsatz” und “Sitzungsaufbau” beeinflusst, die im Folgenden erläutert werden.

Regelinstantiierung. Unabhängig von der jeweilig gewählten Firewall-Architektur - entsprechend den in Kapitel 4 beschriebenen Architekturmöglichkeiten - müssen alle innerhalb des Firewall-Systems verwendeten Filterregeln innerhalb einer Filterkomponente seriell implementiert, modifiziert oder gelöscht werden. Dadurch ist ein Faktor gegeben, der die mögliche Gesamtleistung des Firewall-Systems bestimmt.

Die Zeit T_{Ra}, die benötigt wird eine Regel innerhalb der Filterkomponente zu konfigurieren, ergibt sich aus der Summe der notwendigen Zeit T_{sys} für den Systemaufruf und der Zeit T_{alg} für die Integration der Regel in die Filterdatenbank. Wird als Filterkomponente beispielsweise ein FreeBSD-System mit Filtersoftware ip-filter [104] verwendet, so stellt T_{sys} die Zeit dar, die benötigt wird, einen *ioctl()* Funktionsaufruf auszuführen, um die Regel an den Filter zu übermitteln. Die benötigte Zeit für die Integration einer Regel in die Filterdatenbank ist im Wesentlichen durch die verwendeten Algorithmen und Datenstrukturen [121] bestimmt. Weiterhin wird die Annahme getroffen, dass die Zeiten T_{sys} und T_{alg} für die Konfiguration als auch für das Löschen einer Regel gleich sind $T_{Ra} = T_{Rd} = T_R$. Zusätzlich wird angenommen, das T_R unabhängig vom Zustand der Regeldatenbank ist. Es ergibt sich damit:

$$T_R = T_{sys} + T_{alg} \tag{6}$$

Daraus ergibt sich die maximale Anzahl von Regeloperationen pro Sekunde R_{max}, die innerhalb des Firewall-Systems durchgeführt werden können:

$$R_{max} = \frac{1}{T_R} = \frac{1}{T_{sys} + T_{alg}} \tag{7}$$

Dieser Wert stellt die obere Schranke des Firewall-Systems hinsichtlich der Regelinstantiierung dar.

Sollen durch eine Firewall die in Kapitel 3 beschriebenen Multimedia-Applikationen unterstützt werden, so kann diese obere Schranke folgendermaßen in Bezug zu den zu unterstützenden Applikationen gebracht werden. Eine Multimedia-Sitzung verwendet r dynamisch ausgehandelte Ströme. Für jeden Strom muss jeweils eine Regel innerhalb der Filterkomponente eingetragen und später wieder gelöscht werden. Damit ergibt sich die Anzahl der Regeloperationen pro

Sekunde $R_R = \frac{2r}{T}$, die pro Sitzung durchgeführt werden müssen (T gibt die Dauer einer Sitzung an). Werden über das Firewall-System gleichzeitig n Sitzungen abgewickelt, so ergibt sich unter der Annahme, dass die Sitzungen gleichverteilt sind, folgende Anzahl von Regeloperationen pro Sekunde:

$$R_R = \frac{2rn}{T} \tag{8}$$

Im Bezug zu der oben beschriebenen oberen Schranke ergibt sich folgender Zusammenhang:

$$R_R \leq R_{max} \Rightarrow \frac{2rn}{T} \leq \frac{1}{T_{sys} + T_{alg}} \tag{9}$$

Dadurch kann die Gesamtleistung eines Firewall-Systems unter der Annahme berechnet werden, dass die Regelinstantiierung die Leistungsgrenze bestimmt:

$$N_R = n \leq \frac{1}{2r} \cdot \frac{T}{T_{sys} + T_{alg}} \tag{10}$$

Nimmt man beispielsweise ein H.323-Gespräch ($r = 5$; H.245-Kontrollkanal, zwei RTP- und RTCP-Kanäle für zwei Audioströme; Gesprächsdauer $T = 180\text{sec}$) sowie eine Filterkomponente ($T_{sys} = 5\mu sec$[1], $T_{alg} = 25\mu sec$[2]) so ergibt sich folgende Gesamtleistung:

$$N_R \leq \frac{1}{2 \cdot 5} \cdot \left(\frac{180}{0.000005 + 0.000025} \right) = 600000 \tag{11}$$

Die Beschränkung der Gesamtleistung durch die Regelinstantiierung ist insbesondere für die Klasse der Telefonie-Applikationen von entscheidender Bedeutung. Für Multimedia-Applikationen, die eine deutlich längere Sitzungsdauer besitzen (z.B. VoD-Applikationen) ist eine entsprechend höhere Gesamtleistung möglich. Die durch die Regelinstantiierung gegebene Beschränkung kann insbesondere durch Optimierung von T_{alg} verbessert werden. Dies kann durch Verbesserung der Filterstrukturen erreicht werden [121].

Durchsatz. Die gesamte über die Firewall abgewickelte Kommunikation muss durch die Filterkomponente verarbeitet werden. Dadurch ist die Gesamtleistung eines Firewall-Systems durch den maximalen Durchsatz der Filterkomponente beschränkt.

Der mögliche Durchsatz einer Filterkomponente wird im Wesentlichen durch die Anzahl der zu verarbeitenden Datenpakete bestimmt. Die maximal mögliche Anzahl N_D der gleichzeitig durch das Firewall-System unterstützbarer Multimedia-Sitzungen kann folgendermaßen bestimmt werden:

$$N_D \leq \frac{B(s)}{r \cdot b} \tag{12}$$

Dabei gibt r die Anzahl der Medienströme an, b die für die Medienströme verwendete Bandbreite. Die Bandbreiten der Signalisierungs- und Medienkontrollströme (RTCP) werden vernachlässigt, da diese in der Regel sehr klein gegenüber den Bandbreiten der Medienströme sind. B gibt den maximalen Durchsatz des Paketfilters an, dieser Durchsatz ist von der Paketgröße s abhängig.

Nimmt man beispielsweise ein H.323-Gespräch mit bidirektionaler Audioübertragung ($r = 2$)

[1] Benötigte Zeit für einen *ioctl()* Funktionsaufruf auf einem FreeBSD-System.
[2] Benötigte Filter-Update-Zeiten entsprechend [117].

und eine G.711-Kodierung ($b = 87.2\frac{Kbit}{s}$) an, sowie einen Paketfilter der oberen Leistungsklasse entsprechend [73] mit $B = 2\frac{Gbit}{s}$ bei einer Paketgröße von 218 Byte, so ergibt sich folgende mögliche Gesamtleistung:

$$N_D \leq \frac{2\frac{Gbit}{s}}{2 \cdot 87.2\frac{Kbit}{s}} = 11467 \tag{13}$$

Für Multimedia-Applikationen, die eine wesentlich höhere Bandbreite verwenden (z.B. VoD-Applikationen) ist eine entsprechend niedrigere Gesamtleistung möglich. Die durch den Durchsatz gegebene maximale Gesamtleistung kann insbesondere durch den Entwurf spezialisierter Filterhardware [122] erhöht werden.

Sitzungsaufbau. Eine in einer Firewall-Architektur verwendete Signalisierungsverarbeitung kann nur eine bestimmte Anzahl an Signalisierungsnachrichten pro Sekunde verarbeiten und besitzt damit einen Grenzwert für die Anzahl R_S der Sitzungen, die pro Sekunde aufgebaut werden können. R_S ist im Wesentlichen durch die Anzahl der Nachrichten, die für einen Sitzungsaufbau nötig sind und den für die Verarbeitung der Nachrichten nötigen Aufwand, bestimmt. Über die Dauer T einer Sitzung kann damit, unter der Annahme das die Sitzungen gleichverteilt sind, die mögliche Gesamtleistung des Firewall-Systems wie folgt berechnet werden:

$$N_S \leq R_S \cdot T \tag{14}$$

Durch die in Kapitel 3 vorgeschlagene Verteilung der Firewall, die unter anderem eine Verwendung mehrerer Proxies p ermöglicht kann die Gesamtleistung des Systems erhöht werden.

$$N_S \leq p \cdot \alpha(p) \cdot R_S \cdot T \tag{15}$$

Der Faktor α (efficiency) beschreibt die Tatsache, dass eine lineare Erhöhung der Gesamtleistung nicht möglich ist, da eine vollständige Verteilung nicht erreicht werden kann.

Nimmt man ein monolithisches System entsprechend [24] mit einer Sitzungsaufbaurate von $R_S = 100\frac{1}{s}$, $p = 1$ und $\alpha(1) = 1$ an, sowie eine Sitzungsdauer von $T = 180\text{sec}$ so ergibt sich folgende Gesamtleistung:

$$N_S \leq 18000 \tag{16}$$

Die Gesamtleistung ist wesentlich von der Komplexität der Signalisierung abhängig. Durch eine geeignete Anzahl p und Integration dieser innerhalb des Firewall-Systems kann die Gesamtleistung des Systems verbessert werden (siehe Abschnitt 7.7).

Zusammenfassung. Die oben beschriebenen Faktoren, die die Gesamtleistung eines Firewall-Systems beschränken, müssen in ein ausgewogenes Verhältnis gebracht werden. Nur dadurch kann verhindert werden, dass vorhandene Ressourcen ungenutzt bleiben. Folgendes Beispiel verdeutlicht diesen Sachverhalt: Bei einem Firewall-System, welches über einen hohen Durchsatz verfügt, seine Beschränkung der Gesamtleistung aber durch die Sitzungsaufbaurate erfährt, werden Ressourcen innerhalb der Filterkomponente verschwendet.

Für Firewall-Systeme wird in der Regel (z.B. [24], [73], [122]) der mögliche Durchsatz der Filterkomponente angegeben. Wie oben gezeigt, gibt dieser Faktor aber nicht zwingend Auskunft über die Gesamtleistung des Firewall-Systems.

7.4 Methoden und Werkzeuge zur Leistungsbestimmung

Um alle in Abschnitt 7.3 beschriebenen Leistungskenngrößen erfassen zu können, sind entsprechende Methoden und Werkzeuge notwendig. Wie nachfolgend gezeigt, sind zur Erfassung dieser Größen Messungen notwendig. Zur Erfassung von Leistungskenngrößen in Netzwerken existieren bereits viele Messwerkzeuge, die in Abschnitt 7.4.2 genauer betrachtet werden. Aus dieser Betrachtung folgt, dass für die Bestimmung der notwendigen Firewall-Leistungskenngrößen kein geeignetes Werkzeug zur Verfügung steht. In Abschnitt 7.5 wird deshalb die Spezifikation eines Werkzeugs gegeben, das die Erfassung der notwendigen Kenngrößen ermöglicht. In Abschnitt 7.5.1 wird eine Implementierung dieser Spezifikation vorgestellt, die für die in den darauffolgenden Abschnitten beschriebenen Messungen verwendet wird.

7.4.1 Methoden

Es muss festgelegt werden, auf welchem Weg eine Bestimmung der Leistungskenngrößen eines Firewall-Systems erfolgen kann. Es existieren folgende in Betracht kommende Methoden zur Bestimmung der Kenngrößen: Berechnung, Simulation, Laborexperiment oder Feldstudie.

Für die Bestimmung der Kenngrößen durch eine mathematische Modellbildung fehlen sowohl die entsprechenden notwendigen Modelle der Firewall-Systeme als auch der Multimedia-Applikationen, weshalb diese Methode ausgeschlossen werden muss. Eine Simulation kann ebenfalls nicht durchgeführt werden, da nicht bekannt ist, welche Parameter dafür berücksichtigt werden müssen und wie diese voneinander abhängen. Die Bestimmung der Kenngrößen innerhalb eines realen Szenarios (Feldstudie) ist ebenfalls nicht sinnvoll (Reproduzierbarkeit, Dimensionen, Kontrolle der Variablen).

Ein Laborexperiment kann verwendet werden, wenn die dafür notwendigen Elemente zur Verfügung stehen. Für die Bestimmung der Kenngrößen innerhalb eines Laborexperiments sind folgende Elemente notwendig: Verkehrsgenerator, Messwerterfasser sowie Messobjekt. Im weiteren Verlauf wird genauer auf die ersten beiden Elemente eingegangen. Auf die Beschaffenheit der Messobjekte (der Firewall-Systeme) wird bei den durchgeführten Experimenten in Abschnitt 7.6 und Abschnitt 7.7 eingegangen.

7.4.2 Werkzeuge

Um die Leistungskenngrößen ermitteln zu können, muss durch einen Verkehrsgenerator der notwendige Netzwerkverkehr erzeugt werden. Mit Hilfe eines geeigneten Messwerterfassers müssen dann die zur Bestimmung der Kenngrößen notwendigen Messwerte abgenommen werden.

Verkehrsgenerator. Für die im Kontext dieser Arbeit durchzuführenden Messungen muss der Verkehrsgenerator in der Lage sein, den für eine Multimedia-Applikation charakteristischen Netzwerkverkehr zu erzeugen. Zusätzlich muss der Verkehrsgenerator in der Lage sein, die Kommunikation mehrerer Multimedia-Applikationen gleichzeitig abzuwickeln, damit die zur Messung notwendigen Lastzustände erzeugt werden können. Verfügbare Verkehrsgeneratoren können hinsichtlich der Granularität des erzeugten Verkehrs unterschieden werden.

Die meisten Verkehrsgeneratoren erzeugen Netzwerkverkehr, der über entsprechende Modelle (z.B. stochastische Modelle, Markov-basierte Modelle, selbstähnliche Verkehrsmodelle) berechnet wird. Der so erzeugte Netzwerkverkehr entspricht dann hinsichtlich der innerhalb der Modelle berücksichtigten Parameter (z.B. Paketgröße, Paketrate) dem durch reale Applikationen erzeugten Verkehr. Verkehrsgeneratoren dieser Klasse sind beispielsweise [123] und [124]. Die für die vorliegende Arbeit notwendigen Untersuchungen können nicht mit Hilfe solcher Verkehrsgeneratoren durchgeführt werden, da die Granularität des erzeugten Verkehrs nicht ausreicht, um die zu untersuchenden Effekte innerhalb eines Firewall-Systems hervorzurufen. Beispielsweise wird durch solche Verkehrsgeneratoren nicht (bzw. nur agregiert) die Signalisierungssemantik nachgebildet, wodurch detaillierte Untersuchungen des Signalisierungspfades unmöglich werden.

Als Alternative können Verkehrsgeneratoren verwendet werden, die eine Applikation vollständig beinhalten. Für die Klasse der Multimedia-Applikationen (bzw. Telefonie-Applikationen) sind dies sogenannte Call-Generatoren. Bekannte Call-Generatoren sind beispielsweise [125] oder [126]. Call-Generatoren können mehrere Rufe gleichzeitig abwickeln, für jeden Ruf wird die vollständige Applikationssemantik verwendet (Signalisierung, Protokolle, Zustandsautomaten). Einzig das Verhalten des Benutzers muss durch passende Modelle nachgebildet werden (z.B. Zeitpunkte für Gesprächsinitiierung, Gesprächsdauer).

Messwerterfassung. Die zu erfassenden Leistungskenngrößen müssen durch eine geeignete Messwerterfassung aufgezeichnet werden. Es existieren zwei grundsätzliche Realisierungsmöglichkeiten, die sich durch die Modularität der Lösung unterscheiden.

Die erste Möglichkeit besteht darin, die Messwerterfassung getrennt von dem verwendeten Verkehrsgenerator zu realisieren. Dadurch wird erreicht, dass jeweils sowohl der Verkehrsgenerator als auch die Messwerterfassung ausgetauscht bzw. getrennt voneinander entworfen werden können.

Die zweite Möglichkeit besteht darin, die Messwerterfassung in den Verkehrsgenerator zu integrieren. Das resultierende Messsystem ist dann zwar nicht mehr flexibel, dafür aber in der Regel einfacher umzusetzen und zu verwenden.

Zusammenfassung. Für die vorliegende Arbeit ist es notwendig, als Verkehrsgenerator einen Call-Generator zu verwenden, um alle notwendigen Leistungskenngrößen ermitteln zu können. Verfügbare Call-Generatoren besitzen allerdings wesentliche Nachteile. Softwarebasierte Call-Generatoren sind nicht in der Lage, eine ausreichend große Anzahl paralleler Rufe abzuwickeln. Darüberhinaus ist es nicht möglich, das Verhalten dieser Call-Generatoren ausreichend zu kontrollieren. Das erste Problem kann umgangen werden, indem spezielle hardwarebasierte Call-Generatoren verwendet werden (z.B. [126]). Diese sind in der Lage, die notwendige Leistung zu erbringen. Allerdings besitzen diese den Nachteil, dass sie nicht sehr flexibel sind. Dies gilt einerseits bezüglich der verfügbaren Applikationsprotokolle, andererseits für die Möglichkeit, auf bestimmte Parameter Einfluss nehmen zu können. Zusätzlich liegt der Preis für hardwarebasierte Call-Generatoren etwa um den Faktor 10 über dem für softwarebasierter Call-Generatoren.

7.5 Entwurf und Implementierung eines Messwerkzeuges

Aus den oben genannten Gründen wird im Folgenden der Entwurf die Implementierung und die Verwendungsmöglichkeiten eines softwarebasierten Call-Generators beschrieben, der die oben beschriebenen Beschränkungen nicht besitzt. Für die Überprüfung und Bewertung der in dieser Arbeit getroffenen Aussagen über die Leistung von Firewall-Architekturen wird dieses Messgerät dann verwendet.

Anforderungen. Die Messwerterfassung wird innerhalb des Call-Generators realisiert, um ein möglichst einfach einzusetzendes Messgerät zu erhalten.

Der Call-Generator muss aus mindestens zwei Teilen bestehen, da für den generierten Verkehr eine Quelle (Client) und eine Senke (Server) benötigt wird. Es muss möglich sein, verschiedene Applikationstypen für den Call-Generator zu realisieren. Damit verschiedene Applikationstypen realisiert werden können, muss es möglich sein, die verschiedenen den Applikationen zugrunde liegenden Protokollzustandsmaschinen zu realisieren. Damit diese umgesetzt werden können, müssen einerseits alle Ereignissquellen (*Events*), die einen Zustandswechsel hervorrufen können, innerhalb des Call-Generators verfügbar sein, andererseits müssen die bei einem Zustandswechsel auszuführenden Aktionen (*Actions*) umsetzbar sein.

Folgende *Event*-Quellen müssen berücksichtigt werden:

- **Socket:** Ankommende Netzwerkpakete müssen durch den Call-Generator angenommen werden und können nach ihrer Interpretation einen Zustandswechsel auslösen.
- **Timer:** Durch Ablauf einer gewissen Zeitspanne, kann ein Zustandswechsel ausgelöst werden. Beispielsweise durch einen Timeout nach Senden eines Netzwerkpaketes.
- **User:** Benutzeraktionen können ebenfalls zu einem Zustandswechsel führen. Allerdings werden die bei einer Multimedia-Applikation normalerweise durch den Benutzer ausgeführten Aktionen (z.B. Beenden der Kommunikation) innerhalb des Call-Generators zuvor über die Konfiguration festgelegt und dann in der Regel als Timer abgebildet.

Folgende Aktionen müssen durch das System unterstützt werden:

- **Socket:** Ein Netzwerkpaket muss generiert und verschickt werden.
- **Timer:** Es muss möglich sein, einen neuen Timer zu setzen oder bereits laufende Timer zu modifizieren.

Damit der Verkehrsgenerator in der Lage ist, die Kommunikation mehrerer Applikationen gleichzeitig abzuwickeln, müssen parallel mehrere Zustandsautomaten innerhalb des Call-Generators bedient werden.

Die Messwerterfassung muss die zur Bestimmung der Leistungskenngrößen notwendigen Werte aus der ablaufenden Kommunikation extrahieren und für eine spätere Auswertung aufzeichnen. Damit die zur Auswertung notwendigen Daten an einem Punkt gesammelt werden können, sollte, wenn möglich, die Messwerterfassung entweder nur auf Client-Seite oder nur auf Server-Seite erfolgen.

Folgende Werte müssen erfasst werden:

- **Sitzungsaufbaudauer:** Die Sitzungsaufbaudauer kann auf Client-Seite ermittelt werden, indem die Zeitpunkte bei Erreichen bzw. Verlassen der Zustände innerhalb des Zustandsautomaten der Applikation bestimmt werden, die Beginn bzw. Ende des Sitzungsaufbaus markieren. Die benötigte Zeit für den Wechsel zwischen zwei Zuständen kann dann als Messwert verwendet werden. Auf diese Art können auch weitere benötigte Zeitspannen (Post-Selection-Delay, Answer-Signal-Delay, Sitzungsdauer, Sitzungsabbau), die während dem Ablauf der Kommunikation durchlaufen werden, gemessen werden.
- **Verzögerung:** Um die Verzögerung bestimmen zu können, muss ermittelt werden, wieviel Zeit für den Transport eines Paketes zwischen Client und Server benötigt wurde. Damit die Messwerterfassung innerhalb des Call-Generators erfolgen kann, ist es sinnvoll die zur Bestimmung der Verzögerung notwendige Zeitmarke innerhalb des auszumessenden Paketes selbst zu übermitteln. Damit die Messwerte auf nur einer Seite erfasst werden müssen, können bei Annahme einer bidirektionalen und symmetrischen Kommunikation der Applikation (z.B. Telefonie mit bidirektionaler Audioverbindung) die Pakete in einer Schleife verschickt werden. Die Pakete werden auf Client-Seite mit einer Zeitmarke versehen und an den Server geschickt, dieser sendet dann die Pakete direkt an den Client zurück. Bei Eintreffen eines Paketes kann, durch Bestimmen der aktuellen Zeit und Vergleich mit der im Paket enthaltenen Zeitmarke, die Verzögerung auf dem Transportweg (Hin- und Rückweg) bestimmt werden.
- **Jitter:** Der Jitter kann direkt über die verschiedenen aufgezeichneten Verzögerungsmesswerte der einzelnen Pakete bestimmt werden.
- **Paketverlust:** Um den Paketverlust zu bestimmen, können Sequenznummern innerhalb der Pakete verwendet werden. Auch hier kann bei der Verwendung einer Schleife der Paketverlust für Hin- und Rückweg innerhalb des Clients bestimmt werden.

Die verschiedenen Kenngrößen, insbesondere die Sitzungsaufbaudauer, können nicht pauschal bestimmt werden, da sich hier je nach verwendeter Applikation verschiedene Messpunkte ergeben. Es ist daher nötig, dass die zu erfassenden Kenngrößen für jede zur Messung verwendete Applikation angepasst werden können.

Randbedingungen. Der Call-Generator soll auf gängiger PC-Hardware aufsetzen. Als Betriebssystem wird eine frei verfügbare Software verwendet (Linux/FreeBSD). Dies ist notwendig, um die Kosten des resultierenden Systems gering zu halten. Es ist zu untersuchen, ob mit Hilfe eines PCs die Zeitmessungen in der nötigen Güte ausgeführt werden können.

Die zu messenden Zeitgrößen liegen in einem Bereich, der im Wesentlichen durch die in einem Netzwerk auftretende Verzögerung bestimmt ist. Nimmt man ein 100Mbit Ethernet Netzwerk, sowie die direkte Kommunikation zwischen zwei Rechnern an, so liegt die Verzögerung im Bereich von 100µs bis 250µs. Um in diesem Bereich beispielsweise mit einer Auflösung von 5µs

und einer Genauigkeit von 1%, Messungen durchführen zu können, muss der zur Messung verwendete PC Betriebssystemschnittstellen bereitstellen, die dies ermöglichen.

Ein PC stellt mehrere Mechanismen, mit deren Hilfe ein darauf aufsetzendes Betriebssystem Zeitmessungen durchführen kann, zur Verfügung. Im wesentlichen sind folgende Mechanismen verfügbar: Die Real Time Clock (RTC), der System Timer des Timer Chips (CTC) und der Time Stamp Counter (TSC) der CPU. Innerhalb eines UNIX-basierten PC-Systems wird der System-Timer des CTC für die Taktung des Betriebssystems verwendet. Der CTC wird durch ein 1.1931817MHz Taktsignal gespeist[1]. Durch die Verwendung der Betriebsystemfunktion *gettimeofday()* kann die Zeit, basierend auf dem System Timer des CTC, bestimmt werden. Dadurch kann die Zeit mit einer Auflösung von 0.838µs sowie einer Genauigkeit von 10 ‰ bestimmt werden. Es ist allerdings zu beachten, dass diese Auflösung in der Realität nicht erreicht werden kann, da der Aufruf der Betriebsystemfunktion *gettimeofday()* auf einem PC mehr als 0.838µs benötigt. Der Aufruf der Funktion *gettimeofday()* benötigt auf einem 486DX-33 80µs, auf einem PIII mit 850MHz werden 3.5µs benötigt. Damit sind auf einem modernen PC-System Zeitmessungen mit einer Auflösung von 5µs sowie einer Genauigkeit von 1% gut möglich.

Ein Problem ergibt sich, wenn für einen zu messenden Zeitabstand die beiden dafür notwendigen Zeitpunkte auf unterschiedlichen PC-Systemen genommen werden. Dies ist beispielsweise der Fall, wenn eine Ein-Weg-Verzögerung zwischen Quelle und Senke ermittelt werden muss. In einem solchen Fall müssen die Uhren zwischen beiden an der Messung beteiligten PC-Systemen synchronisiert werden. Dies lässt sich realisieren, indem über das Programm NTPD eine Synchronisation der PC-Uhr mit einem externen Zeitgeber, der für beide PCs verfügbar ist, durchgeführt wird. Als externer Zeitgeber kann das Signal eines GPS-Empfängers verwendet werden. Die Synchronisation der PC-Uhren kann mit einer Genauigkeit von 2µs durchgeführt werden [124]. Diese Präzision ist ausreichend (siehe Abschnitt 7.6 und Abschnitt 7.7), um die im Kontext dieser Arbeit betrachteten Effekte mit Hilfe eines PC-basierten Messwerkzeugs zu untersuchen.

Basisfunktionalität. Um die in Abschnitt 7.6 und Abschnitt 7.7 beschriebenen Messungen durchzuführen, muss eine Multimedia-Applikation ausgewählt werden, die innerhalb des Verkehrsgenerators implementiert wird. Um die Anzahl möglicher Quellen von unerwünschten Effekten (z.B. Implementierungsfehler, Leistungsbeschränkungen) gering halten zu können, wird die Definition eines eigenen, möglichst einfach gehaltenen Multimedia-Protokolls (Test-Protokoll), vorgenommen. Diese Vereinfachung wird erreicht, indem das Test-Protokoll, bzw. die innerhalb des Verkehrsgenerators eingebettete Test-Applikation, nur jene in Kapitel 3 beschriebe-

[1] Durch Division mit 11932 ergibt sich eine Frequenz von etwa 100Hz. Der Kernel des Betriebssystems und damit auch alle dort ablaufenden Aktionen sind auf diese 10.0002ms (genannt *jiffy*) getaktet. Dieser Wert kann durch entsprechende Konfiguration des Kernels in Grenzen verändert werden.

nen Charakteristika verstärkt ausprägt, die wesentliche Auswirkungen auf die Leistung einer Firewall haben. Abbildung 44 stellt die Basisfunktionalität des Test-Protokolls dar:

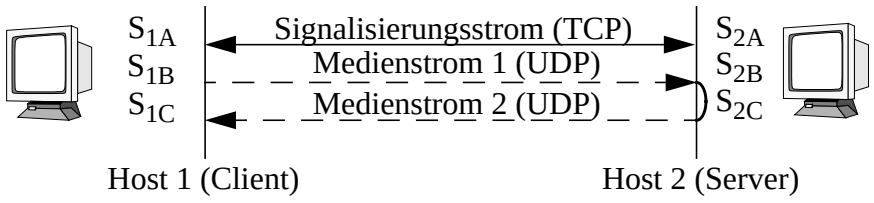


Abbildung 44: Kommunikationsablauf der Test-Applikation

Das Protokoll verwendet einen TCP-Signalisierungskanal, der bei Beginn einer Kommunikation zwischen Client und Server durch den Client aufgebaut wird. Als Medienkanäle werden zwei unidirektionale UDP-Ströme verwendet. Die Mediendaten werden vom Client generiert und an den Server gesendet (Medienstrom 1), der sofort nach Erhalt der Daten diese an den Client zurückgeschickt (Medienstrom 2). Dadurch bekommt die Applikation eine symmetrische Charakteristik, die der Charakteristik einer Telefonie-Applikation (z.B. einer H.323- oder SIP-Applikation) entspricht.

Die resultierende Test-Applikation genügt damit der in Kapitel 2 gegebenen Definition einer Multimedia-Applikation. Folgende Charakteristika einer Multimedia-Applikation besitzt das zur Bestimmung der Leistungskenngrößen verwendete Test-Protokoll:

- **Komplexität der Protokolle:** Es werden verschiedene Teilprotokolle für Signalisierungs- und Medienkanal verwendet. Durch die verwendeten Protokolle muss ebenfalls das Zusammenspiel der einzelnen Ströme organisiert werden.
- **Mehrere Ströme in einer Sitzung:** Es werden verschiedene Kanäle für die Signalisierung sowie die Medien verwendet.
- **Dynamisches Verhalten:** Die Parameter der Medienströme (die verwendeten Portnummern und IP-Adressen) werden vor Verwendung der Medienströme über den Kontrollkanal ausgehandelt.
- **Hohe Datenrate:** Die Datenrate kann angepasst werden, um beispielsweise auch der einer Videoübertragung zu entsprechen.
- **Dienstgüteanforderungen:** Es können Dienstgüteanforderungen für die Medienströme festgelegt werden.

Die Test-Applikation besitzt die in Kapitel 3 gegebenen Applikationscharakteristika nicht. Diese haben im Wesentlichen keine Auswirkungen auf die Leistung einer Firewall. Wie mit diesen Charakteristika in einer Firewall umgegangen werden kann, wurde in Kapitel 6 anhand real verwendeter Applikationen untersucht und dargestellt. Im Folgenden wird auf die Definition des Test-Protokolls eingegangen.

Kommunikationsverhalten. Abbildung 45 beschreibt den Ablauf einer Test-Sitzung, es ist dabei nur der Austausch der Nachrichten auf Applikationsebene dargestellt. Im ersten Schritt wird durch den Client eine TCP-Verbindung zum Server aufgebaut. Über diese TCP-Verbindung wer-

den dann die im Folgenden beschriebenen Signalisierungsnachrichten ausgetauscht. Eine detaillierte Beschreibung der Nachrichtenformate und der einzelnen Nachrichten findet sich in Anhang B.

- Die erste Signalisierungsnachricht wird durch den Client an den Server geschickt. Diese *HELO*-Nachricht enthält die IP-Adresse des Clients sowie optionale Informationen (z.B. Ziel IP-Adresse des Servers), die für ein Call-Routing verwendet werden können. Dadurch kann eine zwischen Client und Server liegende Infrastrukturkomponente die Sitzung bearbeiten und weiterleiten.
- Die empfangene *HELO*-Nachricht des Clients wird durch den Server bestätigt. Der Server sendet eine *ACK*-Nachricht an den Client.
- Der Client sendet nach Empfang der Bestätigung eine *PLAY*-Nachricht an den Server. Innerhalb der *PLAY*-Nachricht gibt der Client die von ihm gewünschte IP-Adresse und Port-Nummer für den Medienstrom 2 an (S_{1C}).
- Der Server antwortet auf diese Nachricht ebenfalls mit einer *PLAY*-Nachricht. Diese *PLAY*-Nachricht enthält zum einen die Bestätigung für die empfangene *PLAY*-Nachricht und zum anderen die vom Server gewünschte IP-Adresse und Portnummer des Medienstroms 1 (S_{2B}).

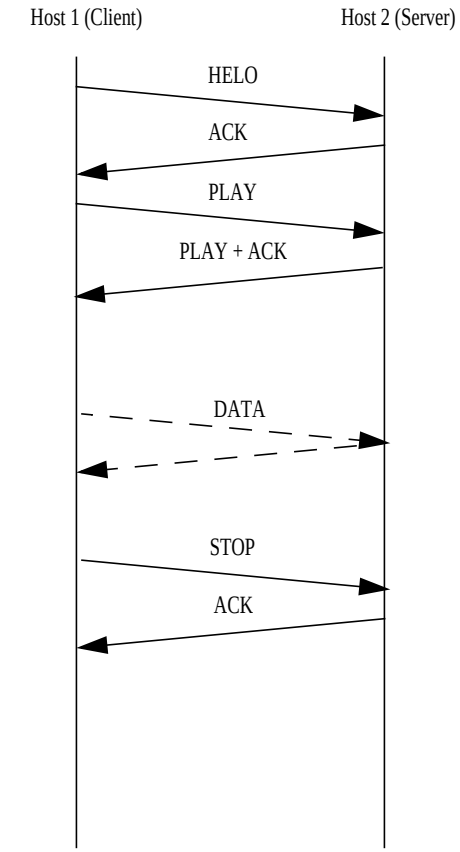


Abbildung 45: Ablauf einer Test-Sitzung

- Nachdem die Medienendpunkte durch Austausch der *PLAY*-Nachrichten spezifiziert sind, beginnt der Client Daten über Medienstrom 1 zu senden. Paketrate sowie Paketgröße müssen zuvor über eine statische Konfiguration festgelegt werden.
- Die übermittelten Daten werden nach Erhalt durch den Server über Medienstrom 2 direkt als Kopie an den Client zurückgeschickt.
- Wenn die Kommunikation abgebaut werden soll, beendet der Client die Medienübertragung und sendet eine *STOP*-Nachricht an den Server.
- Der Server beendet daraufhin ebenfalls die Medienübertragung und bestätigt dann den Erhalt dieser Nachricht.
- Nach Erhalt der Bestätigung ist die Kommunikation zwischen Client und Server beendet.

Für eine Implementierung des oben beschriebenen Kommunikationsverhaltens ist es notwendig, für Client und Server exakte Protokollzustandsmaschinen festzulegen, damit alle möglichen

Kommunikationszustände - und nicht nur der in Abbildung 45 beschriebene Fall - bearbeitet werden können.

Finite State Machine - Client. Abbildung 46 zeigt die zur exakten Definition des Test-Protokolls verwendete Zustandsmaschine auf Seite des Clients.

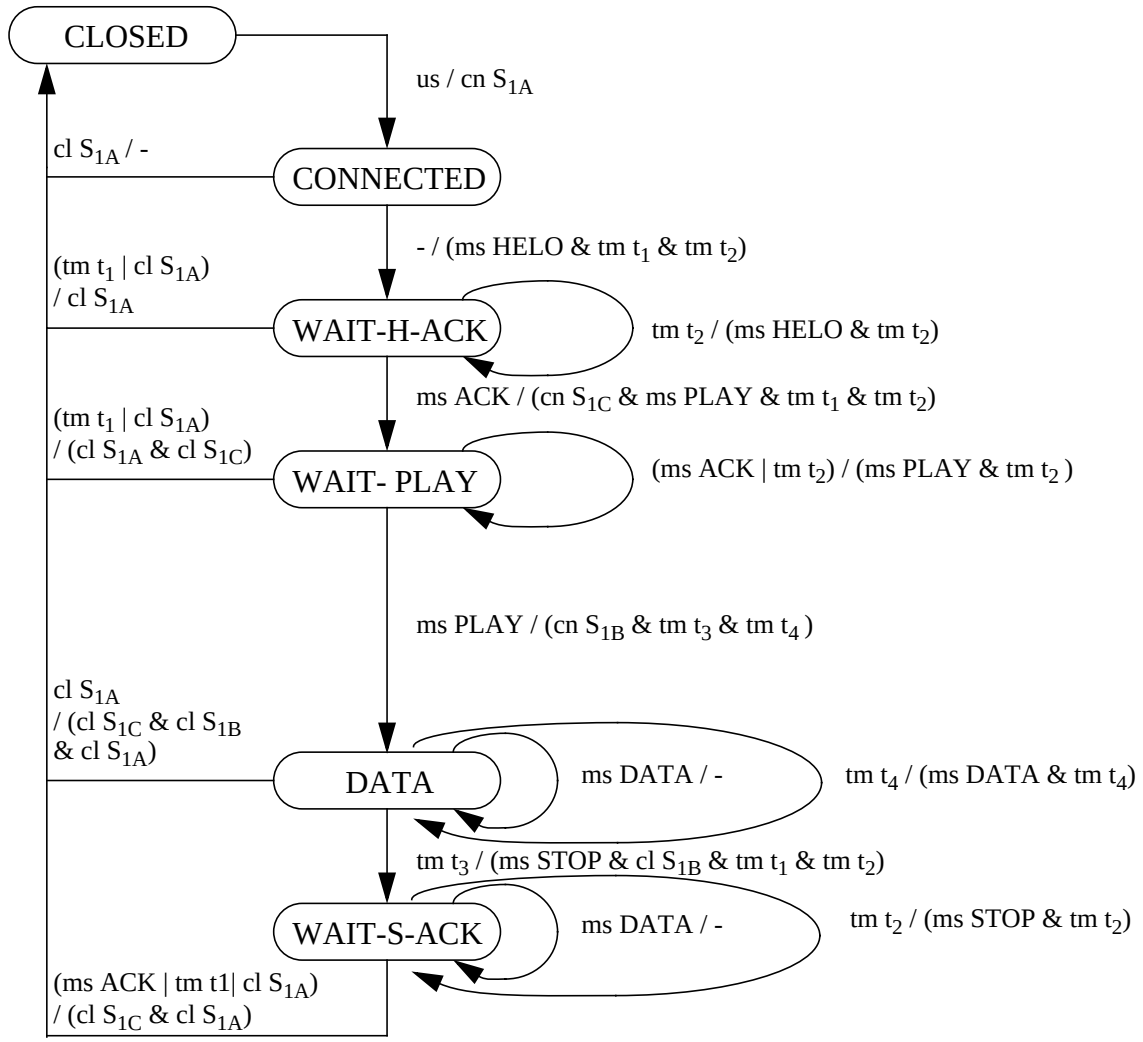


Abbildung 46: Finite State Machine des Test-Protokolls - Client Seite

Die Pfeile beschreiben die Übergänge zwischen den einzelnen Zuständen. Jeder Zustandsübergang ist mit einem *event/action* Paar verknüpft (z.B. us/cn S_{1A}). *Event* beschreibt das den Zustandsübergang auslösende Ereignis. *Action* beschreibt die beim Zustandsübergang auszuführenden Aktionen. Die verschiedenen möglichen Ereignisse, die zum jeweiligen Zustandswechsel führen können sind durch eine Disjunktion (OR = |) markiert. Für jeden Zustandsübergang können wiederum mehrere Aktionen ausgelöst werden. Diese sind durch Konjunktion (AND = &) markiert. Die auszuführenden Aktionen müssen in der Reihenfolge ihrer Auflistung ausgeführt werden. Folgende *Events* treten auf:

- **Socket:**
  cl S_x : Eine Verbindung (assoziiert mit Socket S_x) wurde durch die Gegenseite getrennt (z.B. die TCP-Signalisierungsverbindung wurde geschlossen). Die Bezeichnungen der verwendeten Sockets sind in Abbildung 44 gegeben.
  ms X : Eine Nachricht vom Typ X ist eingetroffen. Die Menge der möglichen Nachrichten ist in Anhang B gegeben.
- **Timer:**
  tm t_x : Timer X ist abgelaufen.
  t_1 : Timeout für alle Wiederholungen einer Nachrichtenübertragung.
  t_2 : Timeout für die Bestätigung einer Nachrichtenübertragung. Wird für eine Nachricht eine Bestätigung erwartet, so muss diese innerhalb der Zeitspanne t_2 eintreffen. Trifft die Bestätigung nicht innerhalb von t_2 ein, wird die Nachricht erneut übertragen. Nachrichtenwiederholungen werden solange ausgeführt bis t_1 abläuft. Es muss daher gelten $t_2 < t_1$ mit $t_1 = nt_2$ für n Nachrichtenwiederholungen.
  t_3 : Timeout für die Medienübertragung. Mediendaten werden solange über Medienstrom 1 verschickt bis t_3 abgelaufen ist.
  t_4 : Timer für die Medienübertragung. Bei jedem Ablaufen von t_4 wird ein Paket über Medienstrom 1 verschickt.
- **User:**
  us : Der Verkehrsgenerator wird über den Benutzer (bzw. Aufgrund des Erreichens eines bestimmten Zustandes des Messsystems) getriggert.

Folgende Aktionen werden verwendet.

- **Socket:**
  cl S_x : Eine Verbindung (assoziiert mit Socket S_x) wird geschlossen.
  cn S_x : Eine Verbindung wird geöffnet.
  ms X : Eine Nachricht des Typs X wird verschickt.
- **Timer:**
  tm t_x : Timer X wird neu gesetzt.

In den folgenden Zuständen kann sich der Client befinden:

- **States:**
  CLOSED : Der Client befindet sich im Ruhezustand; es besteht keine Verbindung.
  CONNECTED : Der TCP-Signalisierungskanal zum Server ist etabliert.
  WAIT-H-ACK : Warten auf die Bestätigung für die initiale *HELO*-Nachricht.
  WAIT- PLAY : Warten auf die PLAY-Nachricht (inklusive ACK für die eigene PLAY Nachricht als piggyback) der Gegenseite.
  DATA: Medienkanäle zwischen Client und Server sind etabliert.
  WAIT-S-ACK : Warten auf eine Bestätigung für das Beenden der Kommunikation.

Finite State Machine - Server. Abbildung 47 zeigt die zur exakten Definition des Test-Protokolls verwendete Zustandsmaschine auf Server-Seite.

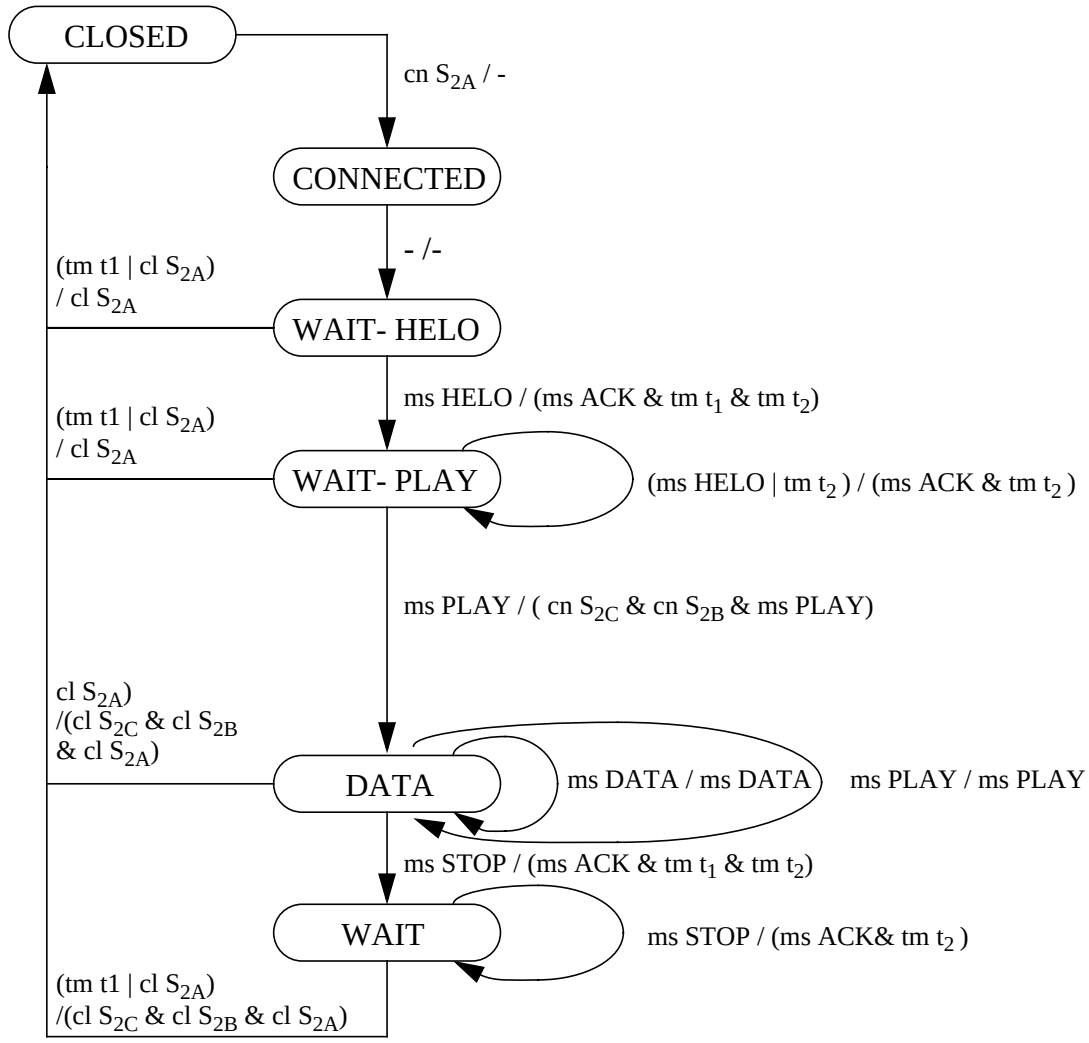


Abbildung 47: Finite State Machine des Test-Protokolls - Server Seite

Es werden die selben *Events* und *Actions* wie auch auf Server Seite verwendet. Die folgenden Zustände können auf Serverseite eingenommen werden:

- **States:**
  CLOSED : Der Serverbefindet sich im Ruhezustand; es besteht keine Verbindung.
  CONNECTED : Der TCP-Signalisierungskanal zum Client ist etabliert.
  WAIT-HELO : Der Server wartet auf die initiale *HELO*-Nachricht.
  WAIT-PLAY : Der Server wartet auf die Medienbeschreibung des Clients.
  DATA : Medienkanäle zwischen Client und Server sind etabliert.
  WAIT : Der Server wartet darauf, dass der Client den Kontrollkanal schließt und damit die Verbindung beendet.

Zeitablaufplan. Die Dauer einer einzelnen Sitzung kann in verschiedene Zeitabschnitte unterteilt werden. Die einzelnen Abschnitte und ihre Reihenfolge sind in Abbildung 48 dargestellt.

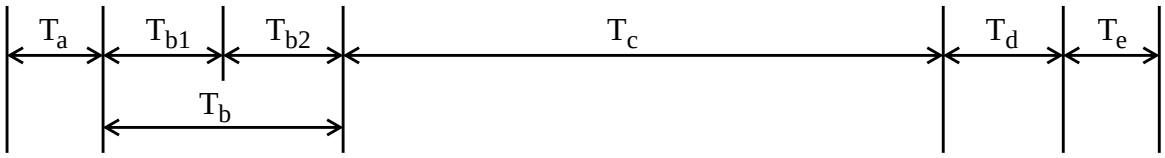


Abbildung 48: Zeitabläufe der Test-Applikation

Die einzelnen Zeitabschnitte haben folgende Bedeutung:

- **T_a :** Zu Beginn einer Test-Sitzung wird der TCP-Signalisierungskanal aufgebaut. T_a gibt die Zeitdauer an, die für den TCP-Verbindungsaufbau benötigt wird.
- **T_b :** Danach werden zwischen Client und Server die Signalisierungsnachrichten ausgetauscht, die für den Sitzungsaufbau benötigt werden. T_{b1} gibt die Zeit an, die benötigt wird, um die initiale *HELO*-Nachricht auszutauschen. T_{b1} ist bestimmt durch die benötigte Zeit, um von dem Zustand *CONNECTED* in der Client-Zustandsmaschine (Abbildung 46) in den Zustand *WAIT-PLAY* zu wechseln. T_{b1} repräsentiert den in Abschnitt 7.2 beschriebenen Rufverzug. T_{b2} ist bestimmt durch die benötigte Zeit, um von dem Zustand *WAIT-H-ACK* in den Zustand DATA zu wechseln. T_{b2} entspricht der in Abschnitt 7.2 beschriebenen Meldedauer.
- **T_c :** Die Dauer des Medientransfers. Sie wird durch den Timer t_3 innerhalb der Client-Zustandsmaschine festgelegt.
- **T_d :** Die Dauer des Sitzungsabbaus. Diese Zeit ist bestimmt durch die benötigte Zeit um von Zustand *DATA* in den Zustand *CLOSED* in der Client-Zustandsmaschine zu erreichen.
- **T_e :** Die benötigte Zeit, um die TCP-Signalisierungsverbindung zu schließen.

Messwerterfassung. Für die Bestimmung der Sitzungsaufbaudauer der Test-Applikation ist es notwendig, die Messgrößen T_a, T_{b1} und T_{b2} zu erfassen. Für die Test-Applikation kann die Sitzungsaufbaudauer T_s entsprechend der Definition in Abschnitt 7.2.1 nach Ablauf der Messung folgendermaßen berechnet werden:

$$T_s = T_a + T_{b1} + T_{b2} \tag{17}$$

Wird innerhalb des Call-Generators eine andere Multimedia-Applikation implementiert, müssen die für die Bestimmung der Sitzungsaufbaudauer verwendeten Messpunkte neu festgelegt werden. Die bei verschiedenen Applikationen gemessene Sitzungsaufbaudauer kann daher auch nur bedingt miteinander verglichen werden.

Die Verzögerung der Mediendaten wird auf Client-Seite ermittelt. Bevor das vom Client gesendete Paket über S_{1B} (siehe Abbildung 44) an den Server gesendet wird, wird die aktuelle Zeit als Zeitstempel T_{cs} innerhalb des Paketes abgelegt (in Anhang B.2 ist die Struktur der Datenpakete angegeben). Auf Server-Seite wird nach Erhalt des Paketes über S_{2B} ebenfalls die aktuelle Zeit als

Zeitstempel T_{sr} in dem erhaltenen Datenpaket abgespeichert. Danach wird das Paket über S_{2C} an den Client zurückgeschickt. Zuvor wird wiederum die aktuelle Zeit als Zeitstempel T_{ss} im Paket abgelegt. Der Client erhält daraufhin das Paket über S_{1C}, die aktuelle Zeit wird danach wiederum als Zeitmarke T_{cr} im Paket abgelegt. Auf Client-Seite kann nun nach Erhalt des Paketes i die Verzögerung $T_{D(i)}$ folgendermaßen berechnet werden:

$$T_{D(i)} = \frac{1}{2} \cdot (T_{cr(i)} - T_{cs(i)} - (T_{ss(i)} - T_{sr(i)})) \tag{18}$$

$T_{D(i)}$ kann auf diese Art nur dann bestimmt werderden, wenn Hin- und Rückweg der Datenpakete die gleichen Eigenschaften besitzen. In den meisten zu untersuchenden Firewall-Szenarien kann aber von symmetrischen Eigenschaften ausgegangen werden (siehe Abschnitt 7.6 und Abschnitt 7.7). Für die oben gegebene Bestimmung von $T_{D(i)}$ müssen die Uhren des Clients und des Servers nicht miteinander synchronisiert werden, was einen einfacheren Messaufbau erlaubt. Die Bestimmung der Zeitmarken auf Server-Seite ermöglicht es, die benötigte Zeit für das Bearbeiten der Daten auf Server-Seite herauszurechnen. Dies gilt insbesondere für den Fall, dass die Zeitmarken innerhalb des Kernels gesetzt werden (siehe unten). Die Messwerterfassung muss für alle I Pakete, die während der Dauer der Sitzung gesendet werden, $T_{D(i)}$ bestimmen und aufzeichnen. Nach Ablauf der Messung kann dann T_D als Mittelwert aller I $T_{D(i)}$ berechnet werden:

$$T_D = \frac{1}{I} \cdot \sum_{i=1}^{I} T_{D(i)} \tag{19}$$

Der Jitter T_j kann aus den einzelnen gemessenen Verzögerungswerten entsprechend der Definition in Abschnitt 7.2.2 nach Ablauf der Messung berechnet werden:

$$T_{J(i)} = \left| T_{D(i)} - T_{D(i+1)} \right| \tag{20}$$

$$T_J = \frac{1}{I-1} \cdot \sum_{i=1}^{I-1} T_{J(i)} \tag{21}$$

Der Paketverlust auf den Medienkanälen kann bestimmt werden, indem die einzelnen Datenpakete eine Sequenznummer erhalten. Auf Client-Seite kann so festgestellt werden, wieviele Pakete verloren gehen. Damit kann der Paketverlust, entsprechend der in Abschnitt 7.2.2 gegebenen Definition, wie folgt berechnet werden:

$$L = \frac{l}{2 \cdot I} \tag{22}$$

Auch diese Bestimmung gilt nur, wenn Hin- und Rückweg der Datenpakete die gleichen Eigenschaften besitzen.

7.5.1 Implementierung des Messwerkzeugs

Die zuvor dargestellte Spezifikation des Messwerkzeuges wurde für verschiedene UNIX Plattformen realisiert. Basis des Messwerkzeuges ist ein handelsüblicher PC, als Betriebssystem wurde Linux bzw. FreeBSD verwendet. Die Implementierung, im Folgenden als *KOMtraffgen* bezeichnet, wurde in C++ vorgenommen.

Architektur. In Abbildung 49 ist die Systemarchitektur schematisch dargestellt. Das System besteht aus zwei Hauptteilen, *Core* und *Application*. Das *Core*-Element stellt das für die Realisierung einer Applikation notwendige Framework bereit. Dazu gehören beispielsweise die für die Realisierung der Applikations-Zustandsmaschine notwendigen Schnittstellen (Timer-Verwaltung, Socket-Abstraktion), die für die Messwerterfassung nötigen Log-Datei Operationen und die für die Parameterisierung der Applikation nötigen Konfigurationsdatei Operationen. Das *Application*-

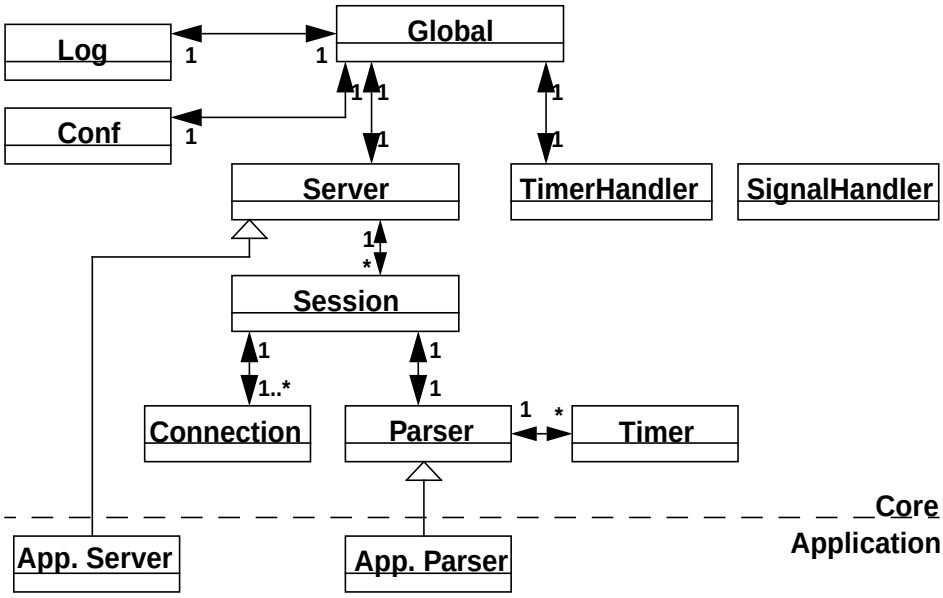


Abbildung 49: *KOMtraffgen*-Systemarchitektur

Element beinhaltet alle applikationsspezifischen Teile. Für jede im Messwerkzeug umzusetzende Applikation ist dieser Teil des Werkzeugs entsprechend anzupassen. Ebenso ist dieser Teil jeweils für die Client und Server Seite zu entwerfen. Die Klasse *ApplicationParser* trägt die Zustandsmaschine der Applikation (Client- oder Server-Seite). Die Klasse *ApplicationServer* ist für die Initiierung (Client-Seite) bzw. das Annehmen (Server-Seite) neuer Sitzungen verantwortlich. Alle Sitzungen werden innerhalb eines Threads verwaltet, jede Sitzung verfügt über ein *Parser*-Objekt (mit Zustandsmaschine und den dafür nötigen Timern) und mehrerer *Connection*-Objekte zur Verwaltung der verwendeten Ströme.

Damit durch ein System möglichst viele Sitzungen unterstützt werden können, wurde zum einen nur ein Thread für die Bedienung aller Sitzungen verwendet (Vermeidung einer Thread-Verwaltung), zum anderen wurde eine Timer-Verwaltung entsprechend [127] verwendet (Reduktion der Komplexität der Timer-Verwaltung).

Parameterisierung der Test-Applikation. Die Parameterisierung erfolgt auf Client-Seite. Folgende Parameter der Test-Applikation müssen über die Konfigurationsdatei an *KOMtraffgen* übergeben werden:

- **Call-Routing:** Für das Call-Routing werden mehrere Parameter benötigt. Es müssen IP-Adresse und Port des Zieles für den TCP-Signalisierungskanal angegeben werden.

Zusätzlich kann die IP-Adresse des Rufziels angegeben werden, falls diese nicht mit dem Ziel der Signalisierung übereinstimmt.

- **Anzahl der Sitzungen** n**:** Dieser Wert legt fest, wieviele Sitzungen gleichzeitig innerhalb von *KOMtraffgen* aktiv sein sollen.
- **Dauer einer Sitzung** T_c**:** Über diesen Wert wird T_c (siehe Abbildung 48) der Sitzung festgelegt.
- **Paketrate** R_P**:** Dieser Wert gibt an, wieviele Pakete pro Sekunde über den Medienstrom 1 verschickt werden sollen.
- **Paketgröße** s**:** Damit wird die Größe der über Medienstrom 1 verschickten Pakete angegeben.

Die Rate R_S, die angibt mit welcher Frequenz neue Sitzungen gestartet werden, wird aus der Anzahl der Sitzungen n sowie der Dauer einer Sitzung T_c berechnet.

$$R_s = \frac{n}{T_c} \tag{23}$$

Dadurch wird erreicht, dass eine konstante Anzahl gleichzeitig ablaufender Sitzungen mit einer homogenen Verteilung der Startzeitpunkte gehalten werden kann.

7.5.2 Eichmessung des Messwerkzeugs

Um später bestimmen zu können, wie die Leistungskenngrößen durch das Einbringen verschiedener Firewall-Systeme in die Kommunikationsstrecke zwischen Client und Server sich ändern, muss zunächst eine Messung ohne Firewall durchgeführt werden. Client und Server wurden auf baugleichen Systemen der unteren Leistungsklasse (FreeBSD 4.5, PIII 850 MHz) realisiert. Zwischen Client und Server wurde ein 100Mbit Full-Duplex Switch verwendet. Die Testapplikation wurde folgendermaßen parameterisiert:

- Die Anzahl der Sitzungen n wird variiert
- Dauer einer Sitzung $T_c = 60s$
- Paketrate $R_p = 50\frac{1}{s}$
- Paketgröße $s = 172byte$

Durch diese Auswahl der Parameter ergibt sich eine Charakteristik der Test-Applikation, die der einer Telefonie-Applikation entspricht. Paketrate und Paketgröße entsprechen denen, die bei Verwendung einer G.711-Audiokodierung ohne Silence Supression verwendet werden.

KOMtraffgen wurde alle 6 Minuten beendet und neugestartet. Bei jedem Neustart wurde die Anzahl der Sitzungen n erhöht, so dass die verschiedenen Leistungskenngrößen in Abhängigkeit der gleichzeitig zu unterstützenden Sitzungen ermittelt werden können. Dadurch kann die Leistungsgrenze des Messwerkzeugs ermittelt werden. Für jedes n wurden dadurch M Messungen durchgeführt (bzw. M Sitzungen wurden durchlaufen).

Für die pro Sitzung ermittelten Kenngrößen T_S, T_D, T_J und L wurde für jedes n der Mittelwert dieser Kenngrößen berechnet.

$$T_S = \frac{1}{M} \cdot \sum_{m=1}^{M} (T_{a(m)} + T_{b1(m)} + T_{b2(m)}) \tag{24}$$

$$T_D = \frac{1}{M \cdot n} \cdot \sum_{m=1}^{M} \sum_{i=1}^{I} T_{D(m,i)} \tag{25}$$

$$T_J = \frac{1}{M \cdot (n-1)} \cdot \sum_{m=1}^{M} \sum_{i=1}^{I-1} T_{J(m,i)} \tag{26}$$

$$L = \frac{1}{M} \cdot \sum_{m=1}^{M} L_{(m)} \tag{27}$$

Sitzungsaufbaudauer:

Abbildung 50 zeigt die gemessene Sitzungsaufbaudauer T_S zwischen Client und Server in Abhängigkeit von n. Es zeigt sich, dass die Leistungsgrenze des Messsystems bei $N = 110$ liegt. Dies entspricht einer Sitzungsaufbaurate von $R_s = 1.83\frac{1}{s}$. Ab diesem Punkt befinden sich 10% mehr Sitzungen im System als durch die Sitzungsaufbaurate festgelegt ist. Die Sitzungen werden schneller erzeugt als sie abgearbeitet werden können. Dieser Schwellwert wird verwendet, um die Leistungsgrenze der untersuchten Systeme zu bestimmen. Für $N > 110$ kann nicht mehr bestimmt werden, wieviel der gemessenen Sitzungsaufbaudauer im Netz (bzw. Messobjekt) anfällt und wieviel durch Verluste innerhalb des Messgerätes entsteht. Die Leistungsgrenze des Messwerkzeugs ist wesentlich durch die verwendete Paketrate R_P bestimmt. Durch Verringerung der Paketrate kann die Leistungsgrenze erhöht werden. Dementsprechend muss bei Variation der *KOMtraffgen*-Parameter der Bereich, in dem das Messwerkzeug sinnvoll eingesetzt werden kann, neu bestimmt werden. Diese Leistungsgrenze gilt natürlich auch für die in der gleichen Messung bestimmten und weiter unten dargestellten Werte für Verzögerung, Jitter und Paketverlust.

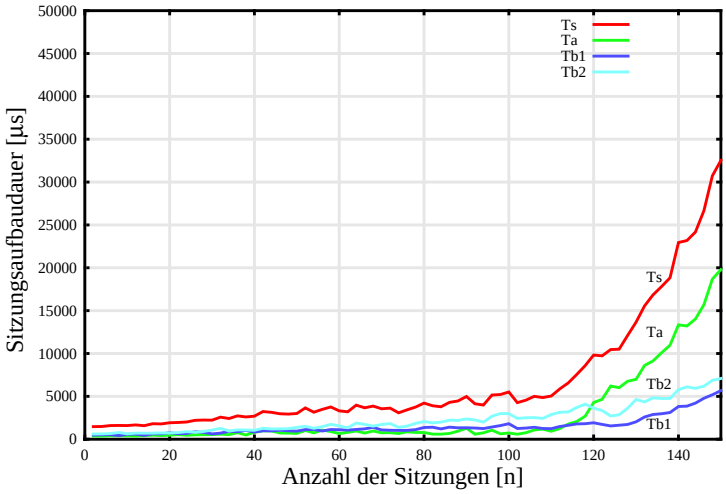


Abbildung 50: Eichmessung: Sitzungsaufbaudauer

Die Varianz (T_S inklusive Standardabweichung ist in Anhang C.1 dargestellt) der Messwerte, bzw. der Variationskoeffizient v (z.B. $v = 45\%$ für $n = 50$) ist sehr hoch. Wird aber ein Messobjekt untersucht, das eine Erhöhung der Sitzungsaufbaudauer von mehreren Millisekunden hervorruft, so verbessert sich der Variationskoeffizient entsprechend. Zum Beispiel ist $v = 2.5\%$ für $n = 50$, bei einer durch das Messobjekt eingefügten Sitzungsaufbaudauer von 50 ms, unter der Annahme, dass durch das Messobjekt nicht ebenfalls die Varianz erhöht wird (siehe dazu Abschnitt 7.6).

Soll bei einer Messung nur der Signalisierungspfad untersucht werden, so kann die weiter unten beschriebene Optimierung *nur Sitzungsaufbau* verwendet werden, um den Variationskoeffizient zu verbessern.

Verzögerung: Abbildung 51 zeigt die gemessenen Verzögerung T_D zwischen Client und Server in Abhängigkeitkeit von n, sowie die Standardabweichung dieser Messwerte. Die hohe Varianz der Messwerte entsteht dadurch, dass zusätzlich zur reinen Netzwerkverzögerung die Zeiten addiert werden, die für das Senden/Empfangen der Pakete benötigt werden (Zeitmessung im User-Space). Der sich ergebende schlechte Variationskoeffizient v (z.B. $v = 60\%$ für $n = 50$) kann, wie bei der zuvor beschriebenen Sitzungsaufbaudauer, verbessert werden, wenn Messobjekte betrachtet werden, die eine hohe Verzögerung verursachen.

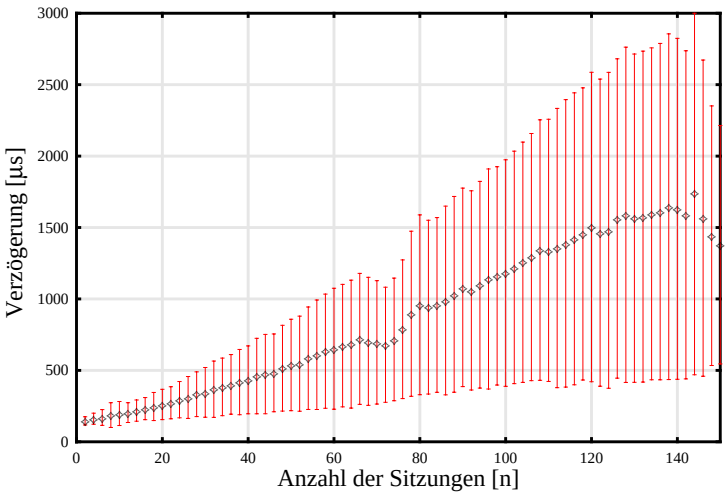


Abbildung 51: Eichmessung: Verzögerung

Soll beispielsweise ein Variationskoeffizient von $v = 5\%$ (für $n = 50$) erreicht werden, so können nur Messobjekte ausgemessen werden, die eine Verzögerung von 5 ms oder mehr verursachen. Um auch Messobjekt, welche nur eine geringe Verzögerung verursachen, betrachten zu können, wurde die weiter unten beschriebene Optimierung *Kernel Stamping* verwendet.

Jitter: Abbildung 52 zeigt den gemessenen Jitter T_J zwischen Client und Server in Abhängkeit von n. Der Jitter nimmt, entsprechend der Varianz des Delays, mit steigendem n zu. Um den Jitter ebenfalls zu verbessern, kann, wie unten beschrieben, ebenfalls die Optimierung *Kernel Stamping* verwendet werden.

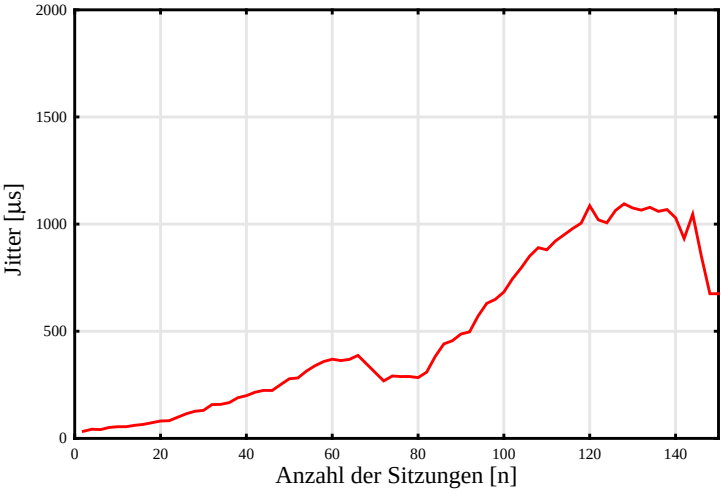


Abbildung 52: Eichmessung: Jitter

Der “Knick” in der Kurve bei $n = 75$ (siehe auch Abbildung 51) ist auf Synchronisationseffekte zurückzuführen. Wird ein Betriebssystem mit einer andereren Taktfrequenz als Träger des Messgerätes verwendet, oder das Meßgerät anders parameterisiert (z.B. Paketrate, Sitzungsdauer) so ver-

schiebt sich die beobachtete Messungenauigkeit. Diese Ungenauigkeit kann verringert werden indem die Optimierung *Kernel Stamping* verwendet wird.

Paketverlust: Bei der hier vorgestellten Messung trat ein Paketverlust von $L = 0\%$ auf. Bei $n = 110$ wird eine Bandbreite von $B = 19.2\frac{Mbit}{s}$ verwendet, dies liegt unterhalb der Netzwerkkapazität von $B = 100\frac{Mbit}{s}$.

7.5.3 Optimierung des Messwerkzeug

Um die durch das Messgerät selbst verursachten Anteile an Sitzungsaufbaudauer, Verzögerung und Jitter zu minimieren bzw. konstant über den betrachteten Messbereich zu halten, werden die im Folgenden dargestellten Optimierungen innerhalb des Messwerkzeuges durchgeführt. So kann vermieden werden, dass anstelle des Messobjektes das Messgerät selbst Gegenstand der durchgeführten Untersuchungen ist.

Optimierung “nur Sitzungsaufbau”. Ist nur der Signalisierungspfad für eine bestimmte Untersuchung von Interesse, so kann über die Konfigurationsdatei des Messwerkzeuges das Senden von Mediendaten abgeschaltet werden. Dadurch werden die notwendigen Netzwerkoperationen (Senden und Empfangen von Daten) innerhalb des Messgerätes deutlich reduziert. Die dadurch reduzierte Interaktion zwischen Applikation und Kernel führt zu einer Glättung der im User-Space gemessenen Sitzungsaufbaudauer.

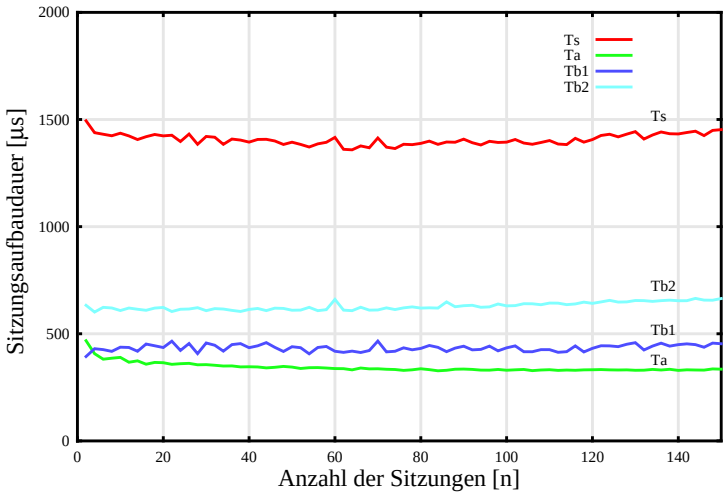


Abbildung 53: Eichmessung: Sitzungsaufbaudauer (optimiert, *nur Sitzungsaufbau)*

Die Varianz (T_S inklusive Standardabweichung ist in Anhang C.2 dargestellt) der Messwerte, bzw. der Variationskoeffizient v (z.B. $v = 11\%$ für $n = 50$) kann so deutlich verbessert werden. Damit ein Variationskoeffizient von $v = 2.5\%$ ($n = 50$) erreicht werden kann, müssen Messobjekte verwendet werden, die eine Sitzungsaufbaudauer von 5 ms oder mehr verursachen.

Optimierung “Kernel Stamping”. Um die im Netzwerk auftretende Verzögerung und den auftretenden Jitter genauer messen zu können kann ein optionales *Kernel Stamping Modul* verwendet werden. Um die im Wesentlichen durch nicht deterministische Kontextwechsel zwischen Applikation und Kernel verursachten starken Schwankungen der Verzögerung zu vermeiden, wurde das unter [103] verfügbare auf ALTQ [105] basierende *Kernel Stamping Modul* verwendet. Die Applikation *KOMtraffgen* konfiguriert über entsprechende *ioctl()* Funktionsaufrufe das *Kernel Stamping Modul*. Über den *ioctl()* Funktionsaufruf wird diesem Modul mitgeteilt, welche Ströme einen Zeitstempel erhalten und an welchem Punkt innerhalb der Pakete dieser anzubringen ist.

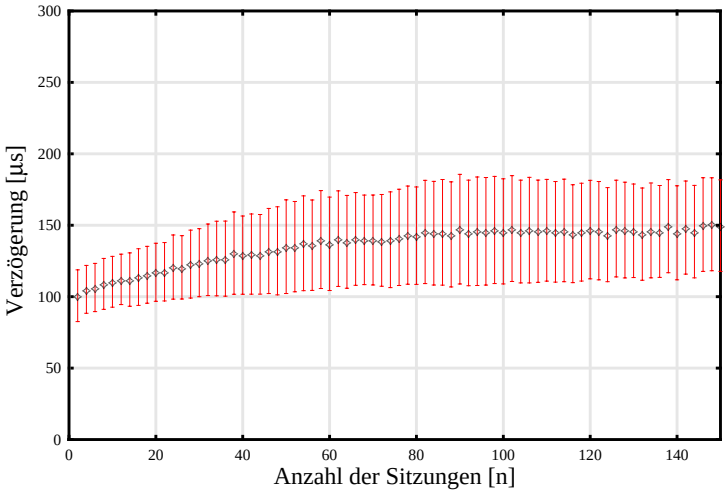


Abbildung 54: Eichmessung: Verzögerung (optimiert, *Kernel Stamping*)

Wird beispielsweise vom Client ein Paket über Medienstrom 1 an den Server geschickt, so wird der Absendezeitpunkt des Paketes nicht vor dem Senden durch *KOMtraffgen* ermittelt und im Paket eingetragen, sondern *KOMtraffgen* lässt den Platz für den Zeitstempel innerhalb des Paketes frei und die Zeitmarke wird durch den Kernel vor Übergabe an die Interfacequeue gesetzt. Dazu muss zuvor dem Kernel über den *ioctl()* Funktionsaufruf mitgeteilt worden sein, dass Pakete dieses Stroms einen Zeitstempel an gewünschtem Offset erhalten. Alle 4 Zeitstempel, die zur Bestimmung der Verzögerung verwendet werden, können über dieses Verfahren gesetzt werden.

Die dadurch erreichte Verbesserung der Verzögerung ist in Abbildung 54 dargestellt, die des Jitters in Abbildung 55. Die zur Messung verwendeten Parameter der Test-Applikation entsprechen denen der zuvor beschriebenen Eichmessung. Der Variationskoeffizient der Verzögerung v (z.B. $v = 23\%$ für $n = 50$) kann damit verbessert werden. Zusätzlich führt die Optimierung zu einer wesentlich geringeren Abhängigkeit der gemessenen Werte vom Lastzustand des Messgerätes bzw vom Messgerät selbst. Die

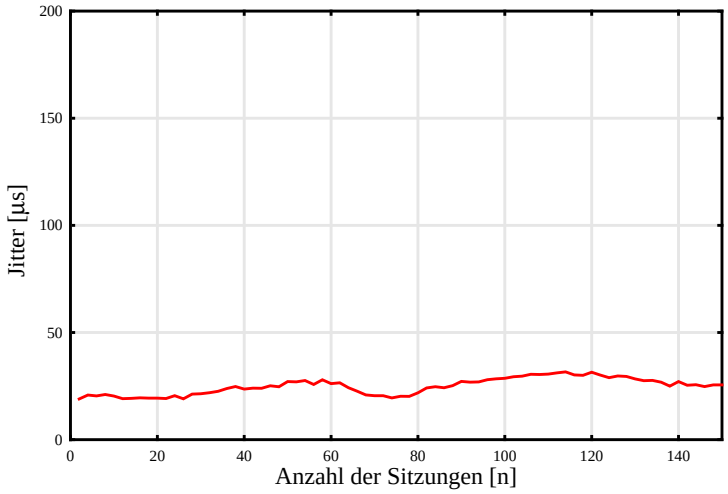


Abbildung 55: Eichmessung: Jitter (optimiert, *Kernel Stamping*)

innerhalb des Messgeräts verursachten Anteile von T_D und T_J können durch Verlagerung der Messpunkte in den Kernel wesentlich reduziert werden. Mit dem optimierten Messgerät können nun kleinere, durch das Messobjekt verursachte Verzögerungswerte bei gleicher Genauigkeit gemessen werden. Soll beispielsweise ein Variationskoeffizient der Verzögerung von $v = 5\%$ ($n = 50$) erreicht werden, so können Messobjekte ausgemessen werden, die eine Verzögerung von 500 µs oder mehr verursachen.

Es ist ebenfalls zu beachten, dass durch die Optimierung für die Messung der Verzögerung und des Jitters die Messung der Sitzungsaufbaudauer sich geringfügig verschlechtert (siehe Anhang C.3). Je nach Ziel der Messung - Sitzungsaufbaudauer oder Verzögerung und Jitter - kann das Messwerkzeug entsprechend verwendet werden.

7.6 Leistungsfähigkeit von Proxy- und Hybridsystemen

In diesem Abschnitt wird gezeigt, dass die Umsetzung des in Kapitel 3 beschriebenen Designprinzips *Trennung von Signalisierungs- und Medienpfad* einer Multimedia-Firewall, neben den in Kapitel 4 dargestellten architekturellen Vorteilen ebenfalls zu einer Verbesserung der Leistungsfähigkeit der resultierenden Firewall führt.

Werden Signalisierungs- und Medienströme in einem Firewall-System getrennt betrachtet, so ermöglicht dies eine getrennte Optimierung des Systems für jeden Strom. In diesem Abschnitt wird betrachtet, wie eine Optimierung hinsichtlich der Dienstgüteparameter der Multimedia-Applikation vorgenommen werden kann. Um zu zeigen, dass eine Optimierung möglich und sinnvoll ist, werden die in Abschnitt 7.3 beschriebenen Leistungskenngrößen, die auf die Dienstgüteparameter der betrachteten Applikation zurückgeführt werden, betrachtet. Dabei werden die Leistungskenngrößen eines normalen Firewall-Systems (nicht optimiert, Fall a) sowie die eines dem Designprinzip folgenden Firewall-Systems (optimiert, Fall b) bestimmt. Durch Vergleiche der gewonnenen Kenngrößen kann gezeigt werden, ob und in welchem Maß die Umsetzung des vorgeschlagenen Designprinzips Auswirkung auf die Leistungskenngrößen hat.

7.6.1 Versuchsaufbau

Die über die Firewall kommunizierenden Applikationen werden durch das im vorigen Abschnitt beschriebene Messwerkzeug *KOMtraffgen* realisiert. Damit wird für die Realisierung der Kommunikationsendpunkte auf jeder Seite der Firewall jeweils ein PC benötigt. Die auf jeder Seite des Firewall-Systems verwendeten Netzwerke wurden über jeweils einen 100Mbit full-duplex Switch realisiert. Abbildung 56 zeigt schematisch den verwendeten Versuchsaufbau.

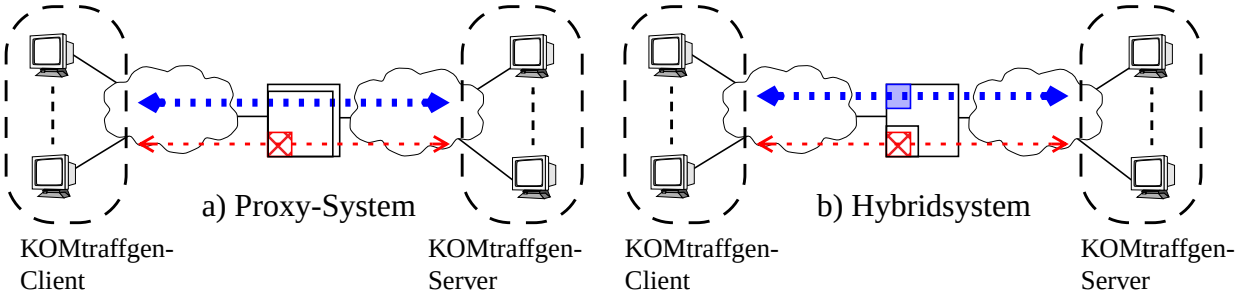


Abbildung 56: Versuchsaufbau I

KOMtraffgen. Das Messwerkzeug *KOMtraffgen* wird für alle im Folgenden durchgeführten Messungen mit den selben Parametern verwendet, damit die gewonnenen Messwerte direkt miteinander verglichen werden können. Als Applikationstyp wurde innerhalb von *KOMtraffgen* die zuvor beschriebene Test-Applikation mit folgender Parameterisierung verwendet:

- Die Anzahl der Sitzungen n wird variiert
- Dauer einer Sitzung $T_c = 60s$
- Paketrate $R_p = 50\frac{1}{s}$
- Paketgröße $s = 172byte$

Es ergibt sich durch diese Auswahl eine Charakteristik der Test-Applikation, die der einer Telefonie-Applikation (G.711-Audiokodierung ohne Silence Supression) entspricht. Die gewählten Parameter entsprechen denen der Eichmessung in Abschnitt 7.5.2. Client und Server werden auf baugleichen PCs realisiert (FreeBSD 4.5, PIII 850 MHz).

Firewall-System a). Das zuerst verwendete Firewall-System ist ein Proxy, der sowohl die Signalisierungsströme als auch die Medienströme verarbeitet. Innerhalb eines Firewall-Systems, das lediglich aus einer Proxy-Komponente aufgebaut ist, werden Signalisierungs- und Medienströme in der gleichen Art und Weise behandelt. Das hier untersuchte Designprinzip *Trennung von Signalisierungs- und Medienpfad* ist demnach in einem solchen Firewall-System nicht umgesetzt. Als Implementierung dieses Firewall-Systems wird das in Kapitel 5 beschriebene *KOMproxyd* Firewall-System mit der Konfiguration "Applikationsmodul *test*, Proxy-Modus; Firewall-Modul *testfw*" verwendet. Verwendet wurde ebenfalls ein FreeBSD 4.5 mit PIII 850 MHz als Trägersystem des Proxies.

Firewall-System b). Als zweites wird ein Hybridsystem verwendet, bestehend aus einem Proxy für den Signalisierungsstrom des Test-Protokolls sowie einem Paketfilter. Die Signalisierungsströme werden durch den Proxy geleitet, die Medienströme werden durch die Filterkomponente geleitet. Zwischen Proxy-Komponente und Filterkomponente besteht dabei eine Interaktionsmöglichkeit (z.B. Übermittlung der Kanalspezifikationen). Damit ist das untersuchte Designprinzip *Trennung von Signalisierungs- und Medienpfad* in diesem Firewall-System umgesetzt. Als Implementierung der Proxy-Komponente wird das in Kapitel 6 beschriebene *KOMproxyd* Firewall-System mit der Konfiguration "Applikationsmodul *test*; Firewall-Modul *testfw*" verwendet. Als Filterkomponente wird das Träger-Betriebssystem des Proxies verwendet. Der Filter besteht dabei nur aus der vorhandenen Routing-Funktionalität des Betriebssystems. Dies entspricht - bezüglich des nötigen Aufwands zur Weiterleitung eines Paketes - einem PC-basierten Paketfilter mit nur einer Regel *allow all* (z.B. ip-filter [104]). Die Kommunikation zwischen Proxy und Filter wird innerhalb des Proxies simuliert. Für die Übermittlung einer Filterregel werden 5µs angenommen. Diese entspricht der notwendigen Zeit eines *ioctl()* Funktionsaufrufs, der normalerweise zur Übermittlung einer Filterregel verwendet wird. Als System wird wieder ein PIII 850 MHz mit FreeBSD 4.5 eingesetzt.

Für die Realisierung beider Firewall-Systeme wurde beidesmal das *KOMproxyd* System verwendet. Dadurch wird (in Grenzen) erreicht, dass anstelle von Implementierungsunterschieden der beiden Firewall-Systeme die Unterschiede der beiden den Firewalls zugrunde liegenden Strategien Ursache für die unterschiedlichen Ergebnisse der Messung sind.

Das zur Messung verwendete Test-Protokoll definiert keine kryptographischen Sicherungsmechanismen für den Signalisierungskanal. Wie in Kapitel 3 beschrieben wurde, ist eine kryptographische Sicherung des Signalisierungsstroms in real verwendeten Szenarien (z.B. H.323- oder SIP-Szenarien) für einen über ein Experimentalstadium hinausgehenden Betrieb zwingend notwendig. Damit die hier durchgeführten Untersuchungen mit realen Szenarien vergleichbar sind, wurde deshalb die Sicherung des Signalisierungskanals innerhalb der *KOMproxyd*-Komponente

(in beiden Firewall-Varianten) simuliert. Für jede auf dem Signalisierungskanal eintreffende Nachricht wird innerhalb des Proxies eine RSA-Signatur Prüfung durchgeführt (Schlüssellänge 512 byte). Auf die Simulation der Generierung einer neuen Signatur nach Bearbeiten der Nachricht wurde verzichtet. Die Verwendung von kryptographischen Mechanismen innerhalb des Signalisierungspfades hat zum einen wesentlichen Einfluss auf die Sitzungsaufbaudauer und zum anderen auf die mögliche Gesamtleistung der Proxykomponente.

7.6.2 Messung

Eichmessung. Um die Leistungskenngrößen der beiden verschiedenen Firewall-Systeme absolut (und nicht nur in Relation zueinander) angeben zu können ist zuvor eine Eichmessung ohne zwischen Client und Server liegende Firewall notwendig. Da die gewählten Parameter und Systeme denen in Abschnitt 7.5.2 entsprechen kann die dort beschriebene Eichmessung als Grundlage für das hier beschriebene Experiment verwendet werden. Aus der Eichmessung ergibt sich ebenfalls, dass das Messgerät nur bis $N = 110$ verwendet werden kann.

Firewall-System a). Das zuerst verwendete Proxy-System stößt bei $N_{fwa} = 64$ Sitzungen an seine Leistungsgrenze. Bei $N_{fwa} = 64$ ist die CPU des Proxies zu 100% ausgelastet. Diese Leistungsgrenze ist im Wesentlichen durch die zu verarbeitenden Medienkanäle bestimmt.

Es ergeben sich die in Abbildung 57 dargestellten Messwerte für die Sitzungsaufbaudauer. Verzögerung und Jitter sind in Abbildung 58 dargestellt.

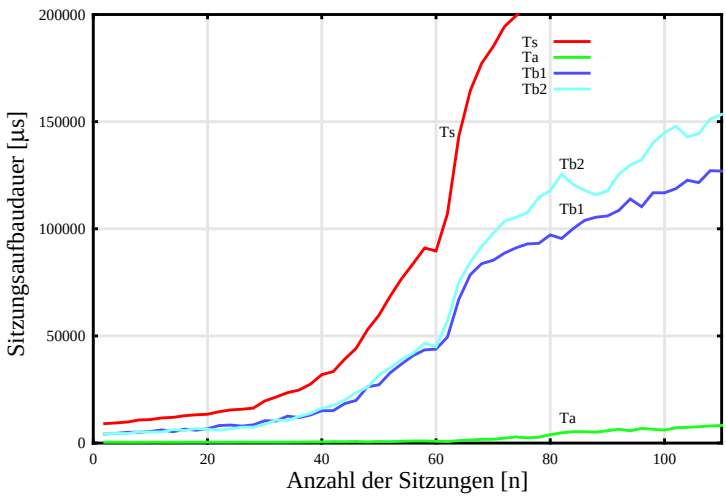


Abbildung 57: I: Sitzungsaufbaudauer Firewall-System a)

Es zeigt sich, dass mit steigender Anzahl gleichzeitig aktiver Sitzungen die gemessenen Werte für Sitzungsaufbaudauer und Verzögerung zunehmen (bezogen auf die durchgeführte Eichmessung). Die Zunahme bis $N_{fwa} = 64$ ist dadurch zu erklären, dass mit steigender Anzahl der Sitzungen mehr Mediendaten durch den Proxy verarbeitet werden müssen. Damit steigt die Wahrscheinlichkeit, dass eine Nachricht (des Signalisierungs- oder Medienstroms) auf die Verarbeitung einer anderen warten muss. Der dann folgende starke Anstieg ab $N_{fwa} = 64$ ist auf die eintretende Überlastung des Systems zurückzuführen.

Die zunehmende Zahl von Sende- und Empfangsoperationen führt ebenfalls zu einer Zunahme der (nicht deterministischen) Kontextwechsel zwischen User-Space und Kernel-Space. Dadurch erhöht sich ebenfalls die Varianz der Verzögerung und somit die des Jitters. Bei eintretendem Überlastfall für $N_{fwa} > 64$ steigt ebenfalls der Paketverlust stark an. Die am Netzwerkinterface

des Systems ankommenden UDP-Pakete können nicht mehr in den Queues des Betriebssystems untergebracht werden und gehen verloren.

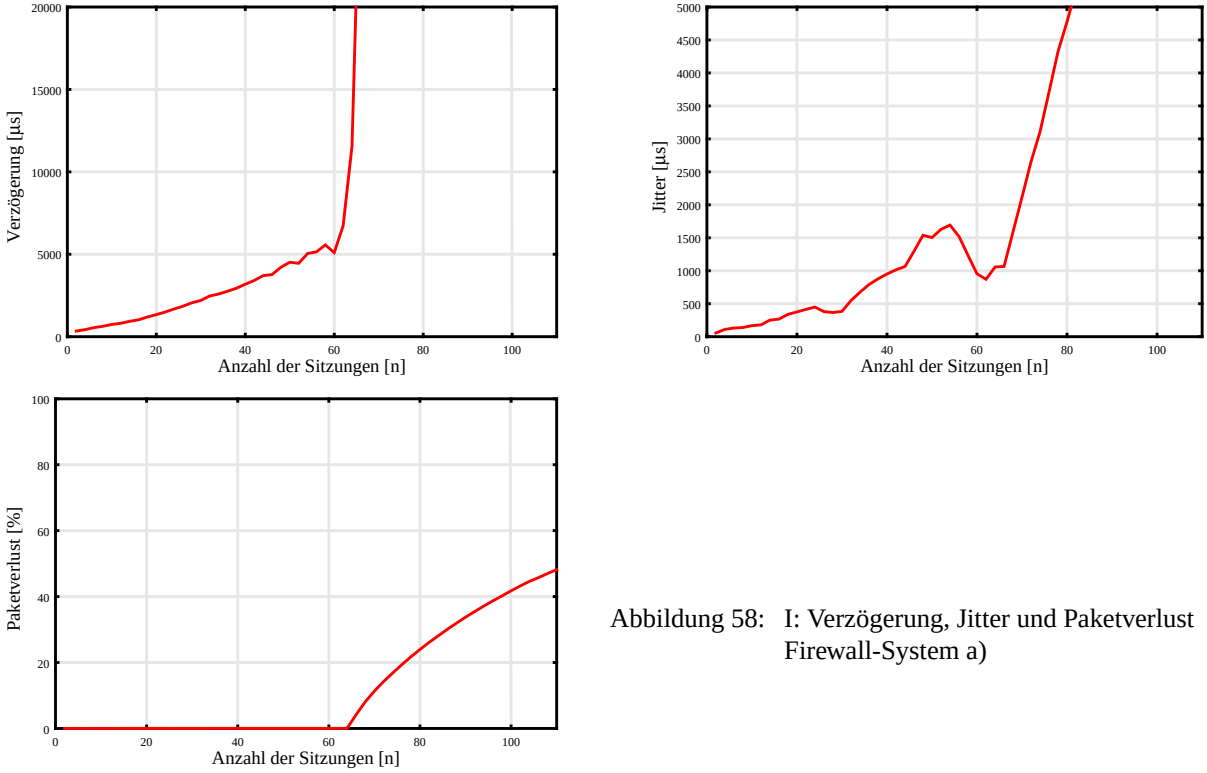


Abbildung 58: I: Verzögerung, Jitter und Paketverlust Firewall-System a)

Firewall-System b). Die Leistungsgrenze des Hybridsystems wurde im Messbereich (n<110) nicht erreicht. Für $n = 110$ wurde eine CPU-Auslastung von 2% erreicht.

Die bei diesem Experiment gemessene Sitzungsaufbaudauer ist in Abbildung 59 dargestellt, Verzögerung und Jitter sind in Abbildung 60 gezeigt.

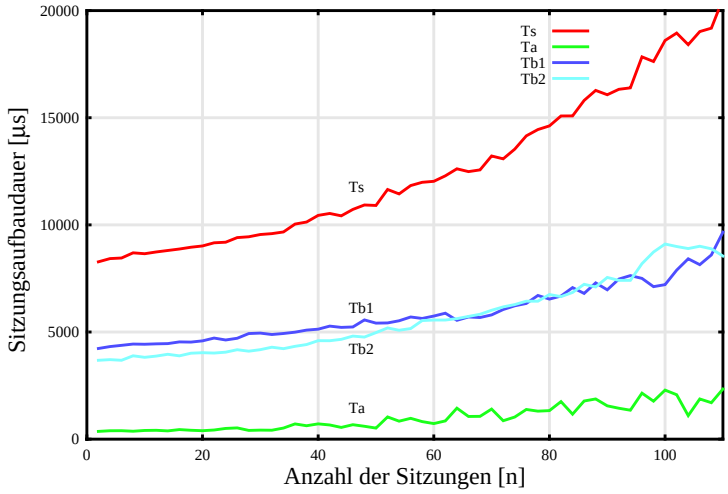


Abbildung 59: I: Sitzungsaufbaudauer Firewall-System b)

Der beobachtete Anstieg der Sitzungsaufbaudauer wird dadurch hervorgerufen, dass mit steigender Anzahl parallel ablaufender Sitzungen die Sitzungsaufbaurate $R_S = n/T_C$ ebenfalls steigt (da die Dauer der Sitzung T_c fix ist). Damit steigt die Wahrscheinlichkeit, dass eine Signalisierungsnachricht auf die Verarbeitung einer anderen warten muss. Auf dieses Problem wird genauer in Abschnitt 7.7 eingegangen.

Verzögerung und Jitter zeigen im betrachteten Messbereich eine sehr geringe Abhängigkeit von der durch das System zu unterstützenden Sitzungsanzahl. Wie weiter unten dargestellt, trägt der

hier verwendete Filter (bzw. Router) einen konstanten Verzögerungsanteil bei.

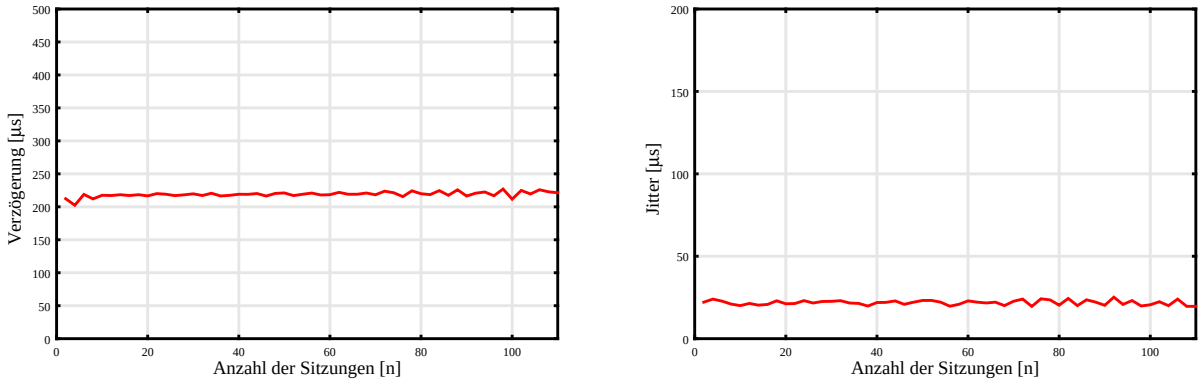


Abbildung 60: I: Verzögerung und Jitter Firewall-System b)

Da keines der an der Messung beteiligten Systeme im betrachteten Messbereich über seine Leistungsgrenze hinaus betrieben wird, wurde auch kein Paketverlust (L=0%) gemessen. Aus diesem Grund ist der Paketverlust bei dieser Messung nicht dargestellt.

7.6.3 Bewertung

Die einzelnen Leistungskenngrößen (siehe Abschnitt 7.3) der beiden untersuchten Firewall-Systeme können nun aufgrund der Messergebnisse bestimmt werden. Die ermittelten Kenngrößen ermöglichen es dann, die untersuchten Systeme zu bewerten und miteinander zu vergleichen.

Im Folgenden wird zunächst anhand der Verzögerung betrachtet, wie die Güte der für die einzelnen Parameter durchgeführten Messungen beurteilt werden kann. Zur Bestimmung einer Leistungskenngröße $G_{T_D}(n)$ muss zunächst $\Delta T_D(n)$ ermittelt werden. Dazu wird von der im jeweiligen Experiment (Messung der Firewall Systeme) bestimmten Verzögerung $T_{Dfw}(n)$ die Verzögerung $T_{D0}(n)$ der Eichmessung subtrahiert:

$$\Delta T_D(n) = T_{Dfw}(n) - T_{D0}(n) \tag{28}$$

Da an jedem Messpunkt beide Verzögerungswerte eine gewisse Varianz besitzen, stellt sich die Frage, in wie weit dem ermittelten Wert $\Delta T_D(n)$ vertraut werden kann. Über die Prüfgröße eines Zweistichproben-t-Tests ist es möglich ein $\Delta T_{D,\alpha}(n) < \Delta T_D(n)$ zu berechnen, so dass mit einer Wahrscheinlichkeit von α angenommen werden kann, dass $\Delta T_{D,\alpha}(n)$ korrekt ist:

$$\Delta T_{D,\alpha}(n) = T_{Dfw}(n) - \left(T_{D0}(n) + t_{2i-2,\alpha} \cdot \sqrt{\frac{s_{fw}^2(n) + s_0^2(n)}{i}} \right) \tag{29}$$

Dabei ist i der Stichprobenumfang an der Stelle n der Experimentalmessung sowie der Eichmessung. Die Stichprobenvarianzen beider Messungen sind mit $s_{fw}^2(n)$ und $s_0^2(n)$ gegeben. Damit kann das Verhältnis $g_{T_D}(n)$ als Maß für die Genauigkeit der Messung verwendet werden:

$$g_{T_D}(n) = 1 - \frac{\Delta T_{D,\alpha}(n)}{\Delta T_D(n)} \tag{30}$$

Für die durchgeführte Bestimmungen der Leistungskenngröße $G_{T_D}(n)$ wird der Maximalwert von $g_{T_D}(n)$ zur Beschreibung der Genauigkeit der Messung verwendet:

$$g_{T_D max} = \max_{0 \le n \le N} \{g_{T_D}(n)\} \tag{31}$$

Analog kann für die anderen gemessenen Kenngrößen ebenfalls ein g_{max} zur Beschreibung der Genauigkeit der Messung angegeben werden. Es ergibt sich aus den obigen Überlegungen:

> *Die Leistungskenngröße $G(n)$ besitzt mit einer Wahrscheinlichkeit von α mindestens die Genauigkeit g_{max}.*

Für die Bestimmung der Genauigkeit wird im Folgenden $\alpha = 0,95$ verwendet. Für die zur Berechnung der Leistungskenngrößen benötigten maximalen Werte für Sitzungsaufbaudauer, Verzögerung, Jitter und Paketverlust wurden die in [119] angegebenen Maximalwerte verwendet. Diese Maximalwerte sind für H.323-basierte Telefonie-Applikationen, eine Übertragung dieser Werte auf die hier verwendete Test-Applikation ist deshalb möglich. Folgende Werte wurden verwendet: $T_{S_{max}} = 4s$, $T_{D_{max}} = 50ms$, $T_{J_{max}} = 10ms$ und $L_{max} = 0.5\%$. Es ergeben sich damit folgende nach (2), (3), (4) und (5) ermittelten Leistungskenngrößen:

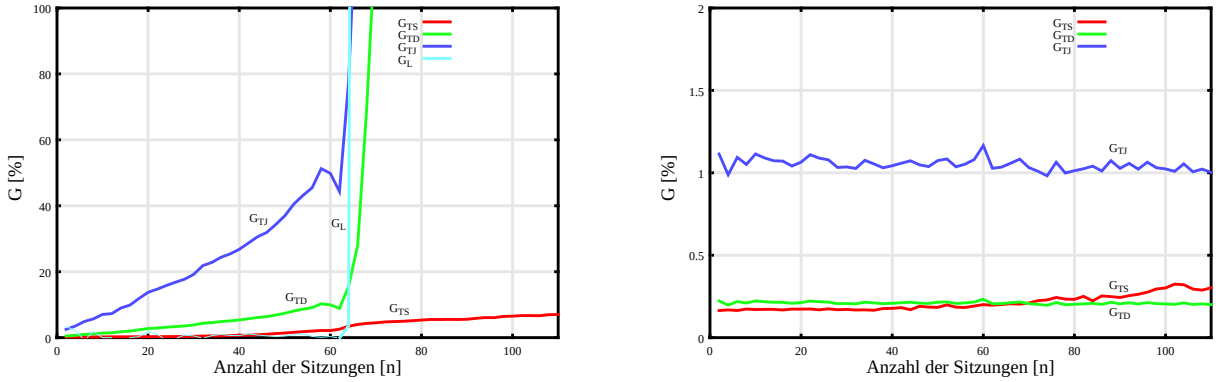


Abbildung 61: I: Leistungskenngrößen von Firewall-System a) und b)

Die Messungen für das Proxy-System wurden mit folgender Genauigkeit durchgeführt: $g_{T_S max} = 8.7\%$, $g_{T_D max} = 2.5\%$. Die Messungen für das Hybrid-System wurden mit folgender Genauigkeit durchgeführt: $g_{T_S max} = 8.2\%$, $g_{T_D max} = 2.0\%$.

Für $g_{T_J max}$ wurde jeweils kein Wert berechnet, die Genauigkeit der Messung des Jitters entspricht der Genauigkeit der Messung der Verzögerung, da beide Werte voneinander abhängen. Für $g_{L max}$ konnte kein Wert berechnet werden, da bei der Eichmessung kein Paketverlust entstanden ist.

Vergleicht man die Leistungskenngrößen beider Firewall-Systeme so wird deutlich, dass der Proxy bei allen Kenngrößen und dabei jeweils für jedes n schlechter abschneidet als das Hybridsystem. Dies gilt auch für die Gesamtleistung, die für den Proxy bei $N_{fwa} = 64$ und für das Hybridsystem mit Sicherheit $N_{fwb} > 110$ außerhalb des betrachteten Messbereiches liegt. Zusätzlich zeigt sich, dass bei dem Proxy-System alle Kenngrößen wesentlich von der Anzahl n der zu unterstützenden Sitzungen abhängen. Die Qualität der über den Proxy abgewickelten Kommuni-

kation hängt damit wesentlich von der Anzahl der gleichzeitig kommunizierenden Endsysteme ab.

Betrachtet man ein realitätsnahes Szenario, in dem die Kommunikationsteilnehmer sich an zwei verschiedenen Standorten befinden und jeweils durch ein Proxy-System geschützt werden, so werden beispielsweise bei $n = 58$ 100% des maximal zulässigen Jitters zwischen den Kommunikationsteilnehmern innerhalb der Firewall-Systeme erzeugt. Dies würde dazu führen, dass alle bei einem Empfänger ankommenden Pakete verworfen werden, da der Jitter auf Empfängerseite nicht mehr kompensiert werden kann.

Damit in einem Anwendungsszenario innerhalb des Netzwerkes bzw. in anderen Netzwerkkomponenten ebenfalls anteilig ein bestimmter Betrag der erlaubten Maximalwerte verwendet werden kann, muss für die Dimensionierung der Firewall ein Maximalwert für die einzelnen Leistungskenngrößen festlegt werden, um die Leistungsgrenze des Systems für ein bestimmtes Szenario festzulegen. Legt man für die oben ausgemessenen Firewall-Systeme ein Maximalwert von 5% für alle Leistungskenngrößen fest, so ergibt sich für das Proxy-System $n = 6$ (bestimmt durch den Jitter) als Leistungsgrenze, das Hybridsystem hingegen kann mindestens im betrachteten Messraum $n < 110$ verwendet werden.

7.6.4 Diskussion

Bei den hier verglichenen Firewall-Architekturen zeigt sich, dass die Verwendung des Prinzips *Trennung von Signalisierungs und Medienpfad* eine wesentliche Optimierung der Leistung eines Firewall-Systems ermöglicht. Zusätzlich zu der hier beschriebenen Leistungssteigerung einer Firewall ermöglicht die getrennte Verarbeitung von Signalisierungs- und Medienströmen die im nächsten Abschnitt beschriebene Verteilung einer Firewall und die dadurch möglichen weiteren Optimierungen. Nicht zu letzt ergeben sich durch die Trennung von Signalisierungs und Medienströmen die bereits in Kapitel 4 beschriebenen Architekturvorteile.

Um bei den dargestellten Experimenten den Unterschied in der Architektur zu vergleichen, und nicht den Unterschied in der Implementierung, wurde für die Realisierung beider betrachteter Firewall-Systeme die gleiche Implementierung verwendet. Beide verwendeten Implementierungen unterscheiden sich lediglich in den Punkten, die zur Umsetzung der verschiedenen Architekturcharakteristika unbedingt verändert werden mussten. Es lassen sich zwar dadurch immer noch nicht die Unterschiede der Architekturen - frei von der Betrachtung spezifischer Implementierungen - in absoluten Zahlen angeben, der dargestellte Unterschied der verschiedenen Architekturen liegt aber in einer solchen Größenordnung, dass mit großer Sicherheit davon ausgegangen werden kann, dass ein signifikanter Anteil der erreichten Verbesserung der geänderten Architektur und nicht der geänderten Implementierung zugesprochen werden kann.

7.7 Leistungsfähigkeit verteilter Firewall-Architekturen

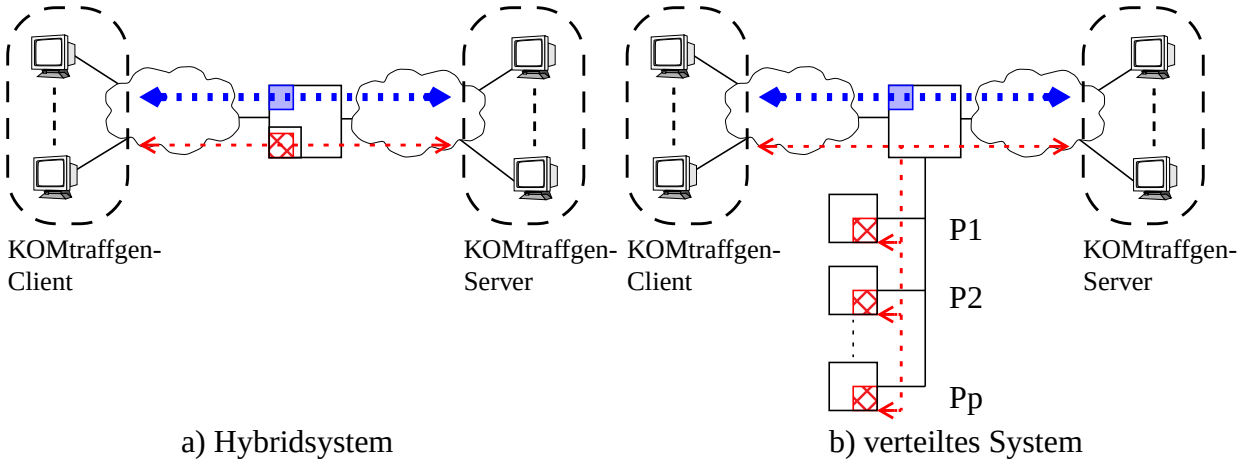
In diesem Abschnitt wird untersucht, welche Auswirkung die Umsetzung des Designprinzips *Trennung von Signalisierungs- und Medienpfad* und die daraus resultierende Möglichkeit der Verteilung (siehe Kapitel 4) auf die Leistungskenngrößen einer Firewall hat. Es wird gezeigt, dass eine lokale Verteilung von (gleichartigen) Firewall-Komponenten einer Firewall zu einer Steigerung der durch die Sitzungsaufbaurate bedingten Gesamtleistung einer Firewall führt.

Die getrennte Betrachtung von Signalisierungs- und Medienströmen ermöglicht es, die verschiedenen Ströme auf unterschiedlichen Wegen durch eine Firewall bzw. ein Firewall-System zu leiten. Dies ermöglicht es, die einzelnen Wege an die Anforderungen des jeweiligen Stromtyps anzupassen (siehe Abschnitt 7.6). Hier wird gezeigt, dass zusätzlich die parallele Verarbeitung der Signalisierungsströme möglich ist und dies zu einer Steigerung (Optimierung) der Gesamtleistung einer Firewall führt.

Für die Untersuchung des Signalisierungspfades wird die den Signalisierungspfad kennzeichnende Leistungskenngröße $G_{T_S}(n)$ bestimmt, sowie die mögliche Gesamtleistung N der Firewall. Es werden zunächst die Leistungskenngrößen eines Hybridsystems (nicht optimiert, Fall a) bestimmt, danach die Kenngrößen einer lokal verteilten Firewall (optimiert, Fall b). Durch die durchgeführten Messungen kann gezeigt werden, dass die durch (15) angegebene Leistungsobergrenze einer Firewall existiert. Durch Vergleich der beiden Systeme kann gezeigt werden, dass diese Leistungsobergrenze durch Verwendung einer lokal verteilten Firewall verbessert werden kann. Der Grad der Parallelisierung in Fall b wird ebenfalls verändert, um den in (15) verwendeten Efficiency-Faktor α zu untersuchen.

7.7.1 Versuchsaufbau

Die über die Firewall kommunizierenden Applikationen werden wiederum durch das Messwerkzeug *KOMtraffgen* realisiert. Für die Realisierung der Kommunikationsendpunkte auf jeder Seite der Firewall wird dafür jeweils ein PC benötigt. Die auf jeder Seite des Firewall-Systems verwen-



a) Hybridsystem b) verteiltes System

Abbildung 62: Versuchsaufbau II

deten Netzwerke werden über jeweils einen 100Mbit full-duplex Switch realisiert, ebenso das DMZ-Netzwerk der verteilten Firewall. Abbildung 62 zeigt schematisch den verwendeten Versuchsaufbau.

KOMtraffgen. Das Messwerkzeug *KOMtraffgen* wird für alle im Folgenden durchgeführten Messungen entsprechend der in Abschnitt 7.5.3 beschriebenen Optimierung *nur Sitzungsaufbau* verwendet. Diese Optimierung kann verwendet werden, da für die hier durchgeführte Untersuchung nur der Signalisierungsweg von Interesse ist. Der Medienweg ist im Wesentlichen für alle in diesem Experiment betrachteten Firewall-Architekturen gleich.

Da in diesem Experiment die durch die Sitzungsaufbaurate R_S gegebene Leistungsgrenze der Firewall-Systeme untersucht wird, wird der Sitzungsaufbaudauer in Abhängigkeit der Sitzungsaufbaurate R_S gemessen. Die Anzahl n der im System gleichzeitig aktiven Sitzungen wird mit $n = 100$ festgelegt, durch Variation der Sitzungsdauer T_c wird die Sitzungsaufbaurate R_S variiert. Diese Art der Messung ermöglicht es große R_S bei geringen n und damit einer geringen Anzahl an Messgeräten zu bestimmen (Bei konstantem $T_c = 180s$ müssten für $R_S = 300\frac{1}{s}$ $n = 54000$ vorgesehen werden, was hier 360 PC-Messsysteme bedeuten würde). Es ergibt sich daraus folgende Parameterisierung von *KOMtraffgen*:

- Dauer einer Sitzung $T_c = \frac{n}{R_S}$
- Es werden keine Medienpakete verschickt.

Aus der durch die Messung bestimmten Funktion $T_S(R_S)$ kann dann die Leistungskenngröße $G_{T_S}(n)$ berechnet werden. Als Applikationstyp wurde innerhalb von *KOMtraffgen* die zuvor beschriebene Test-Applikation verwendet. Es ergibt sich durch diese Auswahl eine Applikations-Charakteristik, die der einer Telefonie-Applikation entspricht. Client und Server werden auf baugleichen PCs realisiert (FreeBSD 4.5, PIII 850 MHz).

Firewall-System a). Das zuerst untersuchte Firewall-System entspricht dem in Abschnitt 7.6 untersuchten und beschriebenen Hybridsystem. Als System wird wieder ein PIII 850 MHz mit FreeBSD 4.5 verwendet. Der Vergleich des Hybrid-Systems mit dem lokal verteilten Firewall System (mit $p = 1$) ermöglicht die Bestimmung des durch die Verteilung der Firewall-Komponenten entstehenden Mehraufwands.

Firewall-System b). Als lokal verteiltes System wurde ebenfalls das Firewall-System *KOMproxyd* verwendet. Auf den einzelnen Proxy-Komponenten wird *KOMproxyd* mit der Konfiguration “Applikationsmodul *test*; Firewall-Modul *remotefw*” verwendet. Die Kanalspezifikationen werden über das DMZ-Netz an die angegebene Komponente per FCP übermittelt. Das verwendete UDP-basierte FCP verwendet 72 Byte große Nachrichten. Zur Ansteuerung der Filter-Komponente wird ebenfalls *KOMproxyd* mit der Konfiguration “Applikationsmodul *fw*; Firewall-Modul testfw” verwendet. Die Filter-Komponente besteht wiederum nur aus der vorhandenen Routing-Funktionalität des Betriebssystems.

Für die Realisierung beider Firewall-Systeme wurde jeweils das *KOMproxyd*-System verwendet, um Implementierungsunterschiede der beiden Firewall-Systeme gering zu halten. Es wurde eben-

falls wie in Abschnitt 7.6 beschrieben, eine kryptographische Sicherung des Signalisierungskanals simuliert.

7.7.2 Messung

Eichmessung. Um die Leistungskenngrößen der beiden verschiedenen Firewall-Systeme absolut angeben zu können, ist zuvor eine Eichmessung ohne zwischen Client und Server liegende Firewall notwendig. Für die Eichmessung wurde zwischen Client und Server ein 100Mbit Full-Duplex Switch verwendet.

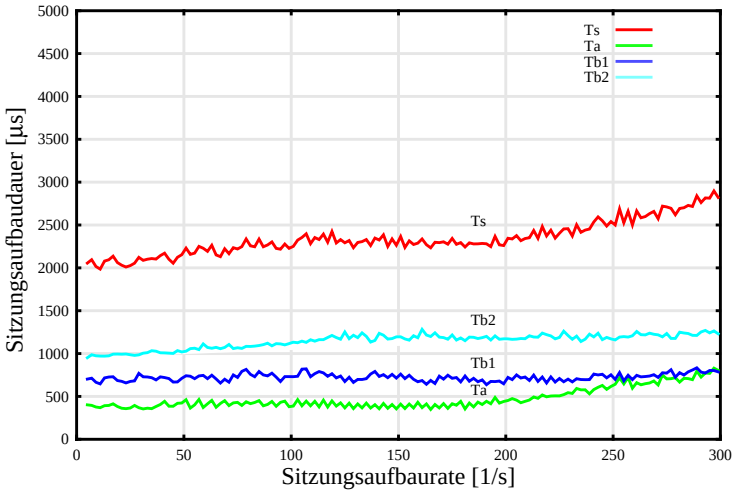


Abbildung 63: II: Sitzungsaufbaudauer (Eichmessung)

Es zeigt sich, dass das Messgerät in dem betrachteten Messbereich $R_S < 300\frac{1}{s}$ nicht seine Leistungsgrenze erreicht. Das Messgerät kann daher im betrachteten Messbereich für die nachfolgenden Messungen verwendet werden. (T_S inklusive Standardabweichung ist in Anhang C.4 dargestellt).

Firewall-System a). Das getestete Hybridsystem erreicht seine Leistungsgrenze bei einer Sitzungsaufbaurate von $R_{Smax} = 88\frac{1}{s}$ (siehe Abbildung 64). Ab diesem Punkt befinden sich 10% ($n > 110$) mehr Sitzungen im System als durch die Setup-Rate festgelegt. Dieser Schwellwert wird verwendet, um die Leistungsgrenze der untersuchten Firewall-Systeme zu bestimmen. Die durch *KOMtraffgen* festgelegte Sitzungsaufbaurate kann durch das Firewall-System ab diesem Punkt nicht mehr bedient werden. In diesem Fall ist die Leistungsgrenze auf das Erreichen der Leistungsgrenze der CPU zurückzuführen. Die Leistungsgrenze kann über $N_{fwa} = T_c \cdot R_{Smax}$ für eine Standard- Gesprächsdauer von $T_c = 180s$ berechnet werden. Diese Berechnung ist unter der Annahme möglich, dass das Systemverhalten von der Sitzungsaufbaurate aber nicht von der Anzahl der Sitzungen abhängig ist. Es ergibt sich dann $N_{fwa} = 15840$. Bis zum Erreichen der

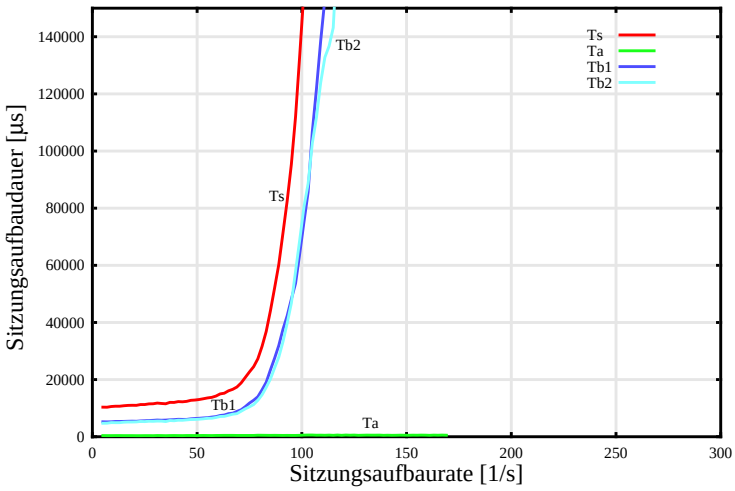
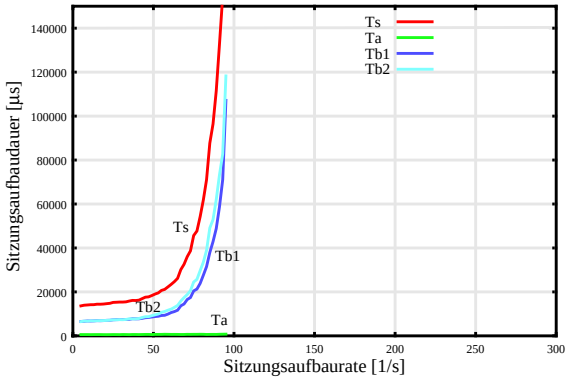


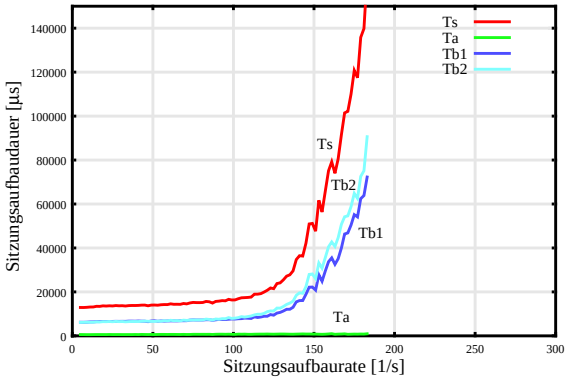
Abbildung 64: II: Sitzungsaufbaudauer Firewall-System a)

Leistungsgrenze kann das Firewall-System die angefragte Sitzungsaufbaurate bei nahezu konstanter Sitzungsaufbaudauer von etwa $T_S = 12ms$ bedienen.

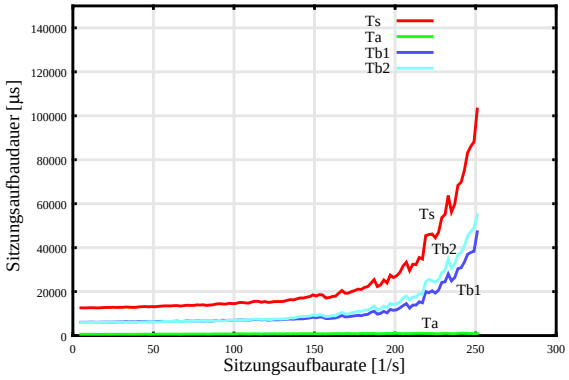
Firewall-System b). Das getestete System erreicht seine Leistungsgrenze (Bestimmung wie bei Firewall-System a)) je nach verwendeter Anzahl der Signalisierungsproxies p bei: $R_{Smax} = 81\frac{1}{s}$ für $p = 1$; $R_{Smax} = 143\frac{1}{s}$ für $p = 2$; $R_{Smax} = 200\frac{1}{s}$ für $p = 3$ (siehe Abbildung 65). Damit können die Leistungsgrenzen für $T_c = 180s$ wie folgt berechnet werden: $N_{fwb1} = 14580$, $N_{fwb2} = 25740$ und $N_{fwb3} = 36000$.



p = 1 Proxy



p = 2 Proxies



p = 3 Proxies

Abbildung 65: II: Sitzungsaufbaudauer Firewall-System b)

Die Leistungsgrenzen sind auch hier bestimmt durch die das Erreichen der Leistungsgrenze der CPUs der einzelnen Proxy-Komponenten. Bei hohen p ist anzunehmen, dass ebenfalls die zur Ansteuerung des Filters verwendete *Firewall Control* an ihre Leistungsgrenze gelangt. Diese Grenze wurde bei den durchgeführten Versuchen nicht erreicht. Bei $p = 3$ lag die CPU-Auslastung dieser Komponente bei etwa 20%. Nimmt man einen linearen Verlauf an, so wird die durch die Leistungsgrenze der *Firewall Control* bestimmte Leistungsgrenze des Gesamtsystems für $p = 15$ erreicht. Ab dieser Grenze kann durch Verteilung keine zusätzliche Steigerung der Gesamtleistung erreicht werden.

Bis zum Erreichen der Leistungsgrenze kann das Firewall-System mit $p = 1$ die angefragte Sitzungsaufbaurate bei nahezu konstanter Sitzungsaufbaudauer von etwa $\overline{T_S} = 16ms$ bedienen. Der Unterschied von etwa 4 ms zwischen dem Hybridsystem und dem verteilten System mit

$p = 1$ wird durch die zwischen *Application Control* und *Firewall Control* aufwendigere Netzwerkkommunikation verursacht. Enthält eine Signalisierungsnachricht einen *PLAY*-Befehl, muss, bevor die Nachricht durch die *Application Control* weitergeleitet wird, die Kanalspezifikation an die *Firewall Control* übermittelt werden und auf eine Bestätigung (ACK) gewartet werden.

7.7.3 Bewertung

Die Leistungskenngröße $G_{T_S}(n)$ kann nun für die einzelnen Firewall-Systeme berechnet werden. Um die Güte der einzelnen Messungen zu beschreiben, wird $g_{T_S max}$ (siehe Abschnitt 7.6.3) angegeben. Zur Bestimmung von $G_{T_S}(n)$ wird $T_{S_{max}} = 4s$ angenommen. Es ergeben sich die folgenden Leistungskenngrößen:

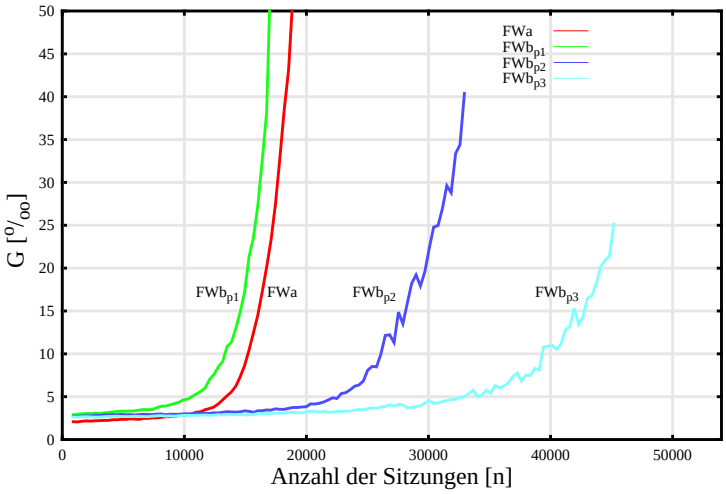


Abbildung 66: II: Leistungskenngrößen von Firewall-System a) und b)

Die Messungen wurden mit folgender Genauigkeit durchgeführt: $g_{T_S maxfwa} = 5.1\%$, $g_{T_S maxfwbp1} = 4.4\%$, $g_{T_S maxfwbp2} = 6.8\%$, $g_{T_S maxfwbp3} = 4.8\%$.

Vergleicht man die Leistungskenngrößen des Hybridsystems und des verteilten Systems mit $p = 1$ so wird deutlich, dass unter dem Gesichtspunkt der Leistungsfähigkeit der Firewall das verteilte System in jeder Hinsicht schlechter abschneidet. Dies gilt sowohl für die Sitzungsaufbaudauer, als auch für die erreichbare Gesamtleistung. Für $p > 1$ wird eine Erhöhung der Gesamtleistung gegenüber dem Hybridsystem erreicht. Allerdings liegt die Sitzungsaufbaudauer dann grundsätzlich, verursacht durch die notwendige Kommunikation der Kanalspezifikationen über das DMZ Netz, über dem des Hybridsystems. Verglichen mit der maximal zulässigen Sitzungsaufbaudauer der hier zugrunde gelegten Telefonie-Applikation spielt dieser Zuwachs für ein reales Szenario aber keine bedeutende Rolle.

Die Gesamtleistung eines Firewall-Systems kann nach (14) berechnet werden. Der Efficiency-Faktor $\alpha(p)$ kann für das hier untersuchte verteilte System aus der jeweils gemessenen Gesamtleistung berechnet werden:

$$\alpha(p = 1) = 1 \Rightarrow \alpha(p = 2) = \frac{25740/2}{14580} = 0.88 \qquad \alpha(p = 3) = \frac{36000/3}{14580} = 0.82$$

Setzt man ebenfalls das Hybridsystem auf die selbe Weise ins Verhältnis, so erhält man hierfür $\alpha = 1.08$. Damit zeigt sich, dass der mit jeder Stufe der Verteilung erreichbare Gewinn geringer ausfällt. Es ist ebenfalls zu beachten, dass p nicht beliebig wachsen kann da, wie beschrieben, ab einem bestimmten p die Leistungsgrenze der Komponente *Firewall Control* erreicht wird.

Wie bereits in Abschnitt 7.3.2 beschrieben, ist es notwendig, alle möglichen oberen Schranken der Gesamtleistung bei der Dimensionierung eines Firewall-Systems zu berücksichtigen, um ein kostenoptimales System zu erhalten. Im Folgenden wird eine Dimensionierung für die hier untersuchten Systeme vorgestellt. Es wird die hier untersuchte Test-Applikation mit einer G.711-Kodierung ($b = 87.2\frac{Kbit}{s}$) und einer Dauer von $T_c = 180s$, sowie einer Filter-Komponente mit $B = 4\frac{Gbit}{s}$ zugrunde gelegt. Durch den Filter ergibt sich damit nach (12) eine obere Leistungsgrenze von $N_D \leq 22935$. Wird zum Ansteuern dieser Komponente das hier untersuchte Hybridsystem verwendet, so erhält man eine durch die Sitzungsaufbaurate gegebene obere Schranke von $N_D \leq 15840$. Damit würde man mehr als 7000 Sitzungen, die noch durch den Filter unterstützt werden können, nicht bedienen können. Anders ausgedrückt, würde dies eine nicht verwendbare Kapazität von $B = 1.2\frac{Gbit}{s}$ innerhalb des Paketfilters bedeuten. Für den angegebenen Filter müsste das verteilte System mit $p = 2$ verwendet werden, um die Verschwendung von Ressourcen zu minimieren. Bei der Dimensionierung wird angenommen, dass die durch die Regelinstantiierung gegebene obere Schranke hier nicht den Engpass darstellt.

7.7.4 Diskussion

Bei den hier verglichenen Firewall-Architekturen zeigt sich, dass die Verwendung einer verteilten Architektur eine wesentliche Steigerung der Gesamtleistung einer Firewall ermöglicht. Es ist daher nicht nur aus den in Kapitel 4 beschriebenen Vorteilen in der Architektur, sondern auch aus dem Gesichtspunkt der Leistungsfähigkeit einer Firewall sinnvoll, verteilte Architekturen einzusetzen.

Es wurde gezeigt, dass die Sitzungsaufbaurate der über eine Firewall laufenden Kommunikation von Multimedia-Applikationen eine obere Schranke der Gesamtleistung eines Firewall-Systems darstellt. Desweiteren wurde gezeigt, dass durch eine lokale Verteilung von (gleichen) Firewall-Komponenten diese Schranke deutlich nach oben verschoben werden kann. Es wurde ebenfalls anhand der untersuchten Implementierungen gezeigt, wie stark der durch eine Verteilung erzielte Gewinn ausfallen kann.

Die in diesem Abschnitt beschriebene und untersuchte obere Schranke für die Gesamtleistung einer Firewall - die Sitzungsaufbaurate - ist von wesentlicher Bedeutung für eine sinnvolle Dimensionierung einer Multimedia-Firewall (siehe obiges Beispiel). Zur Zeit wird von Firewall-Herstellern in der Regel nur die durch den maximalen Durchsatz der Filterkomponente beschriebene obere Schranke beachtet. Wie gezeigt wurde, reicht dies nicht, um die Leistungsfähigkeit einer Multimedia-Firewall ausreichend zu kennzeichnen.

7.8 Zusammenfassung

In diesem Kapitel wurden erstmalig eine Menge aufeinander abgestimmter Leistungskenngrößen definiert, die es ermöglichen die Leistungsfähigkeit von Multimedia Firewalls zu beurteilen. Für die die maximale Gesamtleistung einer Firewall angebende Kenngröße wurde untersucht, welche Faktoren eine obere Schranke für diese Kenngröße festlegen. Es wurde gezeigt, dass nicht nur die maximale Bandbreite und die maximale Geschwindigkeit der Regelinstantiierung der Filterkomponente eine obere Schranke der Gesamtleistung definieren, sondern auch die maximal mögliche Sitzungsaufbaurate. Es wurde gezeigt, dass alle diese Faktoren berücksichtigt werden müssen, um eine Multimedia-Firewall so zu dimensionieren, dass keine Verschwendung von Ressourcen auftritt.

Für die Bestimmung der Leistungskenngrößen wurde ein neues Messwerkzeug entwickelt. Dieses Messwerkzeug *KOMtraffgen* stellt eine leistungsfähige, kostengünstige und flexible Alternative zu kommerziellen Messwerkzeugen dar und ist unter [128] verfügbar. Das System kann, über die in diesem Kapitel beschriebenen Anwendungsgebiete hinaus, für die Bestimmung von Leistungskenngrößen von Komponenten im Kommunikationspfad einer Multimedia-Applikation verwendet werden (z.B. IP-Telefonie Infrastrukturkomponenten).

Anhand der in Kapitel 5 vorgestellten Firewall-Implementierung *KOMproxyd* wurde mit Hilfe des in diesem Kapitel beschriebenen Messwerkzeuges *KOMtraffgen* untersucht, inwieweit sich die in Kapitel 3 vorgeschlagenen Designprinzipien auf die Leistungsfähigkeit einer Firewall auswirken. Es hat sich dabei gezeigt, dass die Verwendung des Designprinzips *Trennung von Signalisierung und Medienpfad* eine wesentliche Optimierung der Leistung eines Firewall-Systems ermöglicht. Durch die Verwendung einer verteilten Firewall-Architektur kann zusätzlich eine wesentliche Steigerung der Gesamtleistung einer Firewall erreicht werden. Zusätzlich wurde untersucht und beschrieben, welcher Grad an Verteilung möglich ist, sowie welche Kosten für eine Verteilung anfallen.

Kapitel 8: Zusammenfassung und Ausblick

Für die produktive Nutzung eines verteilten multimedialen Dienstes ist es von entscheidender Bedeutung, dass dieser nicht nur funktionalen, betrieblichen oder wirtschaftlichen Anforderungen, sondern auch den Sicherheitsanforderungen genügt. Insbesondere Sicherheitsanforderungen, die durch den Einsatz von Firewalls erfüllt werden, konnten bisher im Umfeld multimedialer Dienste nur begrenzt umgesetzt werden.

Die Ergebnisse dieser Arbeit ermöglichen es nun, die durch eine Firewall erbrachten Schutzfunktionen effizient in einem Umfeld zu verwenden, in dem multimediale Dienste verwendet werden. Damit liefert diese Arbeit einen essentiellen Beitrag, um sichere Multimedia-Kommunikation zu realisieren.

8.1 Zusammenfassung

In dieser Arbeit wurde dargelegt, dass die einer Multimedia-Applikation eigenen Charakteristika verantwortlich sind für das Auftreten von Problemen bei dem Zusammenspiel von Multimedia-Applikationen und Firewalls. Ausgehend von einer Klassifizierung dieser Charakteristika wurde dargestellt, dass Veränderung und Erweiterung bestehender Firewall-Architekturen eine geeignete Maßnahme zur Lösung dieser Probleme darstellt.

Es wurde gezeigt, dass das in dieser Arbeit entwickelte Designprinzip *Trennung von Signalisierungs- und Medien-Strömen* die wesentliche Grundlage für die Entwicklung von multimedia fähigen Firewall-Architekturen ist. Die Beachtung dieses fundamentalen Prinzips eröffnet Optimierungsmöglichkeiten, die den Entwurf geeigneter Firewall-Architekturen und Firewall-Komponenten erlauben.

Ausgehend von diesem Prinzip wurde dargestellt, dass verteilte Firewall-Architekturen sich besonders für die Unterstützung von Multimedia-Applikationen eignen. Der Grund hierfür ist, dass diese Architekturen mit den speziellen Charakteristika, insbesondere den Dienstgüteanforderungen einer Multimedia-Applikation besonders gut umgehen können. Dass verteilte Architekturen geeignet für die Unterstützung von Multimedia-Applikationen sind, zeigt sich auch daran, dass andere neuere Forschungsarbeiten diese als Grundlage ihrer aktuellen Untersuchungen verwenden.

In der vorliegenden Arbeit wurde untersucht, welche Mechanismen benötigt werden, um verteilte Firewall-Architekturen realisieren zu können. Bei dieser Betrachtung hat sich gezeigt, dass zum einen effiziente Mechanismen für die Realisierung der Kommunikation zwischen den Komponenten einer verteilten Architektur benötigt werden und andererseits Mechanismen zur Integration der Signalisierungsverarbeitung notwendig sind.

Es wurde dargestellt, dass das Resource Reservation Protocol (RSVP) verwendet werden kann, um die Kommunikation zwischen Firewall-Komponenten einer verteilten Firewall-Architektur zu realisieren. Dadurch ist es möglich, ein bestehendes und erprobtes Protokoll für diese Aufgabe zu verwenden. Anhand einer Implementierung wurde gezeigt, dass RSVP auch in der Praxis für die beschriebene Aufgabe eingesetzt werden kann.

Für die Integration der Signalisierungsverarbeitung wurden entsprechende Lösungsmethoden vorgestellt. Es wurde verdeutlicht, dass die für die Integration der Signalisierungsverarbeitung notwendigen Mechanismen von bestehenden Infrastrukturkomponenten abgeleitet werden müssen. Durch eine praktische Implementierung der gefundenen Mechanismen konnte gezeigt werden, das diese auch real eingesetzt werden können.

Besonderes Augenmerk wurde auf die Leistungsfähigkeit von Firewall-Architekturen gerichtet. Es wurden in dieser Arbeit neue Leistungskenngrößen definiert, die es ermöglichen, die Leistungsfähigkeit von Multimedia-Firewalls zu bestimmen. Es wurden Messungen an verschiedenen Firewall-Architekturen durchgeführt, die zeigen, dass verteilte Firewall-Architekturen auch hinsichtlich ihrer Leistungsfähigkeit am besten geeignet sind, um Multimedia-Applikationen zu unterstützen. Es wurde ebenfalls beschrieben, welche Faktoren berücksichtigt werden müssen, um eine Multimedia-Firewall für bestimmte Leistungsanforderungen richtig dimensionieren zu können.

Im Rahmen dieser Arbeit wurden verschiedene Werkzeuge entwickelt, die für die Überprüfung der getroffenen Aussagen verwendet wurden, aber auch über die Arbeit hinausgehend in Zukunft verwendet werden können. Das Werkzeug *KOMtraffgen* stellt ein Messwerkzeug zur Bestimmung von Leistungskenngrößen dar. Es kann verwendet werden um Leistungskenngrößen von Komponenten im Kommunikationspfad einer Multimedia-Applikation zu bestimmen. Das Werkzeug *KOMproxyd* kann verwendet werden, um Firewall-Architekturen für Multimedia-Applikationen umzusetzen. Dieses Werkzeug wird beispielsweise durch das DFN innerhalb des DFN-Videokonferenzdienstes praktisch verwendet.

8.2 Ausblick

Damit multimediale Dienste sinnvoll genutzt werden können, muss den strikten Dienstgüteanforderungen der Multimedia-Applikation Rechnung getragen werden. Diese Arbeit hat sich bei der Betrachtung dieses Aspektes darauf beschränkt, Methoden zu finden, die dazu führen, dass die Dienstgüteparameter (Delay, Jitter, Loss) möglichst gering und vorhersagbar beeinflusst werden. Außerdem wurden Methoden entwickelt, die es ermöglichen, die Gesamtleistung einer Firewall zu erhöhen.

In zukünftigen Netzen wird es der Fall sein, dass vor einer ablaufenden Kommunikation Vereinbarungen über die einzuhaltende Dienstgüte getroffen werden. In diesem Fall ist es dann notwendig, dass alle auf dem Kommunikationsweg liegenden Komponenten sich an die getroffenen Vereinbarungen halten. Die Verwendung einer Firewall, die entlang des Kommunikationspfades liegt, darf nicht dazu führen, dass an dieser Stelle die Vereinbarungen nicht eingehalten werden. Um dies zu verhindern, müssen Wege gefunden werden, eine Firewall in die Aushandlung und

Umsetzung von Dienstgütevereinbarungen mit einzubeziehen. Die in dieser Arbeit gezeigte Verwendung des RSVP-Protokolls zur Steuerung einer Firewall stellt einen vielversprechenden Startpunkt zur Lösung dieses in Zukunft verstärkt auftretenden Problems dar.

Diese Arbeit hat sich mit einem speziellen Baustein zur Durchsetzung von Sicherheitsanforderungen beschäftigt: der Firewall. Es hat sich aber gezeigt, dass einzelne Bausteine zur Realisierung von IT-Sicherheit nur näherungsweise singulär betrachtet werden dürfen. Sicherheit kann nur dann erreicht werden, wenn alle Bausteine mit ihren Wechselwirkungen in ihrer Gesamtheit betrachtet werden. Zur theoretischen sowie praktischen Betrachtung dieser Gesamtheit fehlen geeignete Methoden, Modelle und Werkzeuge oder diese sind nur eingeschränkt verwendbar. Eine wesentliche Aufgabe der Wissenschaft ist es, diese Lücke in der Zukunft zu schließen, damit komplexe und dennoch sichere Systeme geschaffen werden können.

Literaturverzeichnis

[1] M. Schumacher, U. Roedig, and M. L. Moschgath. *Hacker Contest. Sicherheitsprobleme, Lösungen, Beispiele* . Springer Verlag, Heidelberg, Germany, September 2002. ISBN 3-540-41164-X.

[2] D. Döerner. *Die Logik des Misslingens. Strategisches Denken in komplexen Situationen*. Rowohlt Verlag, 1989. ISBN 3-499-19314-0.

[3] Bundesamt für Sicherheit in der Informationstechnik. *Kosten und Nutzung der IT-Sicherheit*. SecuMedia Verlags GmbH, Ingelheim, Germany, 2000. ISBN 3-922746-34-9.

[4] S. Fischer, C. Rensing, and U. Roedig. *Open Internet Security - Von den Grundlagen zu den Anwendungen*. Springer Verlag, Heidelberg, Germany, January 2000. ISBN 3-540-66814-4.

[5] J. Scambray, S. McClure, and G. Kurtz. *Hacking Exposed: Network Security Secrets & Solutions*. Osborne/McGraw-Hill, Berkeley, U.S.A., 2001. ISBN 0-07-212748-1.

[6] J. Allen. *The CERT Guide To System and Network Security Practices*. Addison-Wesley, May 2001. ISBN 0-201-73723-X.

[7] C. Busch and S. Wolthusen. *Netzwerksicherheit. A Vorkosgan Adventure*. Spektrum Akademischer Verlag, 2002. ISBN 3-827-41373-7.

[8] C. Eckert. *IT-Sicherheit: Konzepte - Verfahren - Protokolle*. Oldenbourg, 2001. ISBN 3-486-25298-4.

[9] W. Cheswick and S. Bellovin. *Firewalls and Internet Security: Repelling the Wily Hakker*. Addison-Wesley, 1994. ISBN 0-201-63357-4.

[10] E. D. Zwicky, S. Cooper, D. B. Chapman, and D. Russell. *Building Internet Firewalls (2nd Edition)*. O'Reilly & Associates, 2000. ISBN 1-56592-871-7.

[11] J. Crowcroft, M. Handley, and I. Wakeman. *Internetworking Multimedia*. Morgan Kaufmann Publishers, 1999. ISBN 1-55860-584-3.

[12] International Telecommunication Union. Visual Telephone Systems and Equipment for Local Area Networks which provide a non-guaranteed Quality of Service. *Recommendation H.323, Series H: Audiovisual and Multimedia Sytems. Telecommunication Standardization Sector of ITU, Geneva, Switzerland*, May 1996.

[13] C. Rensing, U. Roedig, R. Ackermann, and R. Steinmetz. A Survey of Requirements and Standardization Efforts for IP-Telephony-Security. In *Proceedings of the Workshop "Sicherheit in Netzen und Medienströmen"*, pages 50–60, September 2000. ISSN 1431-472-X.

[14] International Telecommunication Union. Security and encryption for H-Series (H.323 and other H.245-based) multimedia terminals. *Recommendation H.325, Series H: Audiovisual and Multimedia Sytems. Telecommunication Standardization Sector of ITU, Geneva, Switzerland*, November 2000.

[15] International Telecommunication Union. Baseline Security Profile. *Recommendation H.235 Annex D, Series H: Audiovisual and Multimedia Sytems. Telecommunication Standardization Sector of ITU, Geneva, Switzerland*, November 2000.

[16] International Telecommunication Union. Signature Profile. *Recommendation H.235 Annex E, Series H: Audiovisual and Multimedia Sytems. Telecommunication Standardization Sector of ITU, Geneva, Switzerland*, November 2000.

[17] International Telecommunication Union. Hybrid Security Profile. *Recommendation H.235 Annex F, Series H: Audiovisual and Multimedia Sytems. Telecommunication Standardization Sector of ITU, Geneva, Switzerland*, March 2002.

[18] *Meyers großes Taschenlexikon, Band 20*. Bibliographisches Institut & F.A. Brockhaus AG, 1987. ISBN 3-411-11005-8.

[19] Organisation for Economic Co-Operation and Development. *OECD Guidelines for the Security of Information Systems and Networks - Towards a Culture of Security.* Organisation for Economic Co-Operation and Development, 2002.

[20] R. Shirey. Internet Security Glossary. RFC 2828, Internet Engineering Task Force, May 2000.

[21] N. Pohlmann. *Firewall-Systeme*. MITP-Verlag, Bonn, Germany, 2000. ISBN 3-8266-4075-6.

[22] K. Egevang and P. Francis. The IP Network Address Translator. RFC 1631, Internet Engineering Task Force, May 1994.

[23] Check Point. CheckPoint Firewall-1. http://www.checkpoint.com.

[24] Cisco Systems. Cisco PIX. http://www.cisco.com.

[25] U. Ellermann and C. Benecke. Parallele Firewalls - skalierbare Lösungen für Hochgeschwindigkeitsnetze. In *DFN-CERT Workshop Sicherheit in vernetzten Systemen, Hamburg*, March 1998.

[26] R. Steinmetz. *Multimedia-Technologie: Grundlagen, Komponenten und Systeme*. Springer Verlag, October 2000. 3. Auflage (erstmalig mit CD).

[27] U. Roedig. Firewalls and their Impact on Multimedia Systems. Multimedia Computing and Networking 2000, January 2000. Panel Discussion "Security Firewalls and their Impact on Multimedia Systems".

[28] M. Shore. H.323 and Firewalls: Problem Statement and Solution Framework. Internet Draft, Internet Engineering Task Force, February 2000. Work in progress.

[29] Intel. H.323 and Firewalls: The problems and pitfalls of getting H.323 safely through firewalls. White paper, Intel, April 1997.

[30] C. Rensing, U. Roedig, R. Ackermann, L. Wolf, and R. Steinmetz. VDMFA, eine verteilte dynamische Firewallarchitektur für Multimedia-Dienste. In *Tagungsband Kommunikation in verteilten Systemen (KiVS), Darmstadt, Germany*, pages 144–157. Springer, March 1999. ISBN 3-540-65597-2.

[31] U. Roedig, R. Ackermann, C. Rensing, and R. Steinmetz. A Distributed Firewall for Multimedia Applications. In *Proceedings of the Workshop "Sicherheit in Netzen und Medienströmen", Berlin*, pages 3–16, September 2000. ISSN 1431-472-X.

[32] U. Roedig, R. Ackermann, C. Rensing, D. Rohrdrommel, J. Schlesinger, and R. Steinmetz. Distributed Firewall for Multimedia Applications. Patent Registration, June 2000.

[33] D. Senie. Network Address Translator (NAT)-Friendly Application Design Guidelines. RFC 3235, Internet Engineering Task Force, January 2002.

[34] Microsoft. Developing NAT Friendly Applications. White paper, Microsoft, 2000.

[35] R. Falk. Web-Anwendungen und Firewalls. In *Proceedings of 6. Deutscher IT-Sicherheitskongreß*, pages 29–43. SecuMedia Verlags GmbH, Ingelheim, Germany, May 1999.

[36] R. Falk. Java RMI, CORBA und Firewalls. In *Proceedings of Java Informationstage 1998, JIT'98*, pages 215–223. Springer, November 1998.

[37] International Telecommunication Union. Packet based Multimedia Communication Systems. *Recommendation H.323v4, Series H: Audiovisual and Multimedia Sytems. Telecommunication Standardization Sector of ITU, Geneva, Switzerland*, November 2000.

[38] International Telecommunication Union. Call signalling protocols and media stream packetization for packet-based multimedia communication systems. *Recommendation H.225.0, Series H: Audiovisual and Multimedia Sytems. Telecommunication Standardization Sector of ITU, Geneva, Switzerland*, September 1999.

[39] International Telecommunication Union. Isdn user-network interface layer 3 specification for basic call control. *Recommendation Q.931, Series Q: Switching and Signalling. Standardization Sector of ITU, Geneva, Switzerland*, May 1998.

[40] International Telecommunication Union. Control protocol for multimedia communication. *Recommendation H.245, Series H: Audiovisual and Multimedia Sytems. Telecommunication Standardization Sector of ITU, Geneva, Switzerland*, June 2000.

[41] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: a transport protocol for real-time applications. RFC 1889, Internet Engineering Task Force, January 1996.

[42] R. Steinmetz, R. Ackermann, U. Roedig, M. Goertz, and M. Schumacher. IP-Telefonie: Protokolle, Herausforderungen, Loesungen und kritische Analyse der Sicherheit. Presentation Series - Arbeitsgemeinschaft des VDE Rhein-Main, January 2002.

[43] R. Ackermann, M. Schumacher, U. Roedig, and R. Steinmetz. Vulnerabilities and Security Limitations of current IP Telephony Systems. In *Proceedings of the Conference on Communications and Multimedia Security (CMS 2001), Darmstadt*, pages 53–66, May 2001.

[44] M. Steinebach, F. Siebenhaar, C. Neubauer, R. Ackermann, U. Roedig, and J. Dittmann. Intrusion Detection Systems for IP Telephony Networks. In *REAL TIME INTRUSION DETECTION (LA DETECTION DES INTRUSIONS EN TEMPS REEL) - a Symposium organised by the INFORMATION SYSTEMS TECHNOLOGY (Estoril, Portugal 27-28 May 2002)*, May 2002.

[45] U. Roedig, R. Ackermann, M. Tresse, L. Wolf, and R. Steinmetz. Verbesserte Systemsicherheit durch Kombination von IDS und Firewall. In *Systemsicherheit*, pages 117–128, March 2000. ISBN 3-528-05745-9.

[46] European Telecommunications Standards Institute. H.323 Threat Analysis. *Draft, Telecommunications and Internet Protocol Harmonization over Networks, ETSI*, March 2000.

[47] M. Handley, H. Schulzrinne, E. Schooler, and J. Rosenberg. SIP: Session Initiation Protocol. RFC 2543, Internet Engineering Task Force, March 1999.

[48] R. Fielding, J. Gettys, J. Mpdgiö, H. Frysyk, and T. Berners-Lee. HTTP: Hypertext Transfer Protocol – HTTP/1.1. RFC 2068, Internet Engineering Task Force, January 1997.

[49] J. Postel. Simple Mail Transfer Protocol. RFC 821, Internet Engineering Task Force, August 1982.

[50] M. Handley and V. Jacobson. SDP: Session Description Protocol. RFC 2327, Internet Engineering Task Force, April 1998.

[51] Cisco Systems. Security in SIP-Based Networks. White paper, Cisco Systems, San Jose, California, July 2002.

[52] M. Thomas. SIP Security Requirements. Internet Draft, Internet Engineering Task Force, November 2001. Work in progress.

[53] J. Undery, S. Sen, and V. Torvinen. Enhanced Usage of HTTP Digest Authentication for SIP. Internet Draft, Internet Engineering Task Force, June 2002. Work in progress.

[54] J. Rosenberg and H. Schulzrinne. Session Initiation Protocol (SIP): Locating SIP Servers. RFC 3263, Internet Engineering Task Force, June 2002.

[55] C. Griwodz, M. Zink, M. Liepert, G. On, and R. Steinmetz. Multicast for Savings in Cache-based Video Distribution. In *Proceedings of SPIE's Multimedia Computing and Networking Conference 2000 (MMCN'00), San Jose, USA*, pages 26–35. SPIE, January 2000.

[56] Real Networks. Real Networks. http://www.real.com.

[57] P. Lehti. Konzeption und Entwicklung eines RTSP-Firewall Moduls. Studienarbeit, April 2001. KOM-S-0104.

[58] H. Schulzrinne, A. Rao, and R. Lanphier. Real Time Streaming Protocol (RTSP). RFC 2326, Internet Engineering Task Force, April 1998.

[59] R. Frederick. RTSP Bakeoff. http://www.dmn.tzi.org/ietf/mmusic/48/slides/mmusic-rtsp-bakeoff.PDF.

[60] M. Zink, A. Jonas, C. Griwodz, and R. Steinmetz. LC-RTP (Loss Collection RTP): Reliability for Video Caching in the Internet. In *Proc. of 7th Int'l Conf on Parallel and Distributed Systems (ICPADS): Workshops*, pages 281–286. IEEE, Piscatay Way, NJ, USA, July 2000. ISBN 0-7695-0571-6.

[61] OpenH323. OpenGatekeeper H.323 Proxy. http://openh323proxy.sourceforge.net.

[62] Real Networks. RTSP Proxy Kit 2.0. http://www.rtsp.org/2001/proxy.

[63] P. Srisuresh, J. Kuthan, J. Rosenberg, A. Molitor, and A. Rayhan. Middlebox Communication Architecture and Framework. Internet Draft, Internet Engineering Task Force, March 2002. Work in progress.

[64] M. Leech, M. Ganis, Y. Lee, R. Kuris, D. Koblas, and L. Jones. SOCKS protocol version 5. RFC 1928, Internet Engineering Task Force, March 1996.

[65] P. M. Gleitz and S. M. Bellovin. Transient Addressing for Related Processes: Improved Firewalling by Using IPV6 and Multiple Addresses per Host. In *Proceedings of the Eleventh Usenix Security Symposium*, August 2001.

[66] B. Schneier. *Secrets & Lies: Digital Security in a Networked World*. John Wiley & Sons. ISBN 0-471-25311-1.

[67] U. Roedig. H.323 und Firewalls, Probleme und Lösungen. Studie, Deutsches Forschungsnetz (DFN), February 2002.

[68] U. Roedig, R. Ackermann, and R. Steinmetz. IP-Telefonie und Firewalls, Probleme und Lösungen. *Praxis in der Informationsverarbeitung und Kommunikation (PIK)*, 1(24):32–40, January 2001.

[69] OpenH323. OpenH323 Protocol Stack. http://www.openh323.org/.

[70] R. Knobbe, A. Purtell, and S. Schwab. Advanced security proxies: an architecture and implementation for high performance network firewalls. In *In Proceedings of DARPA information survivability conference and exposition 2000, Vol 1*, pages 140–148, 2000.

[71] S. Mercer, A. Molitor, M. Hurry, and T. Ngo. H.323 Firewall Control Interface (HFCI). Internet Draft, Internet Engineering Task Force, December 1998.

[72] A.Molitor. Firewall Control for IP Telephony. http://www.aravox.com.

[73] ARAVOX. ARAVOX 1000. http://www.aravox.com.

[74] R. Swale, P. Mart, P. Sijben, S. Brim, and M. Shore. Middlebox Communications (MIDCOM) Protocol Requirements. Internet Draft, Internet Engineering Task Force, November 2001. Work in progress.

[75] M. Barnes. Middlebox Communications (MIDCOM) Protocol Evaluation. Internet Draft, Internet Engineering Task Force, July 2002. Work in progress.

[76] J. Rosenberg, J. Weinberger, C. Huitema, and R. Mahy. STUN - simple traversal of UDP through network address translators. Internet Draft, Internet Engineering Task Force, July 2002. Work in progress.

[77] M. Gaynor and S. Bradner. Firewall Enhancement Protocol (FEP). RFC 3093, Internet Engineering Task Force, April 2001.

[78] S. Ioannidis, A. Keromytis, S. Bellovin, and J. Smith. Implementing a Distributed Firewall. In *Proceedings of Computer and Communications Security (CCS)*, pages 190–199, November 2000.

[79] A. Papp. Firewall Redundancy Protocol Specification. Internet Draft, Internet Engineering Task Force, June 2000. Work in progress.

[80] J. Xu and M. Singhal. Design and Evaluation of a High-Performance ATM Firewall Switch and Its Applications. *IEEE JSAC*, 17(6):1190–1200, June 1999.

[81] J. T. McHenry, P. W. Dowd, T. M. Carrozzi, F. A. Pellegrino, and W. B. Cocks. An FPGA-Based Coprocessor for ATM Firewalls. In K. L. Pocek and J. Arnold, editors, *IEEE Symposium on FPGAs for Custom Computing Machines*, pages 30–39, Los Alamitos, CA, 1997. IEEE Computer Society Press.

[82] G. Montenegro and V. Gupta. Sun's SKIP Firewall Traversal for Mobile IP. RFC 2356, Internet Engineering Task Force, June 1998.

[83] R. Finlayson. IP Multicast and Firewalls. RFC 2588, Internet Engineering Task Force, May 1999.

[84] J. Quittek. Using SNMP as Midcom Protocol. Internet Draft, Internet Engineering Task Force, April 2002. Work in progress.

[85] J. Renkel. Evaluation of RSIP against MIDCOM requirements. Internet Draft, Internet Engineering Task Force, May 2002. Work in progress.

[86] S. Sen, C. Aoun, and T. Taylor. Applicability of MEGACO to Middlebox Control. Internet Draft, Internet Engineering Task Force, May 2002. Work in progress.

[87] T. Taylor. Evaluation Of DIAMETER Against MIDCOM Requirements. Internet Draft, Internet Engineering Task Force, April 2002. Work in progress.

[88] C. Aoun et al. COPS applicability as the MIDCOM protocol. Internet Draft, Internet Engineering Task Force, May 2002. Work in progress.

[89] M. Stiemerling and J. Quittek. Simple Middlebox Configuration (SIMCO) Protocol Version 1.0. Internet Draft, Internet Engineering Task Force, July 2002. Work in progress.

[90] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin. Resource ReSerVation Protocol (RSVP) – Version 1 Functional Specification. RFC 2205, Internet Engineering Task Force, September 1997.

[91] L. Zhang, S. Deering, D. Estrin, S. Shenker, and D. Zappala. RSVP: a new resource ReSerVation protocol. *ieeenet*, 7(5):8–18, September 1993.

[92] M. Stiemerling and J. Quittek. MIDCOM Protocol Semantics. Internet Draft, Internet Engineering Task Force, June 2002. Work in progress.

[93] S. Herzog. RSVP Extensions for Policy Control. RFC 2750, Internet Engineering Task Force, January 2000.

[94] S. Yadav, R. Yavatkar, R. Pabbati, P. Ford, T. Moore, and S. Herzog. Identity Representation for RSVP. RFC 2752, Internet Engineering Task Force, January 2000.

[95] F. Baker, B. Lindell, and M. Talwar. RSVP Cryptographic Authentication. RFC 2747, Internet Engineering Task Force, January 2000.

[96] S. Kent and R. Atkinson. Security Architecture for the Internet Protocol. RFC 2401, Internet Engineering Task Force, November 1998.

[97] L. Berger and T. O'Malley. RSVP Extensions for IPSEC Data Flows. RFC 2207, Internet Engineering Task Force, September 1997.

[98] L. Berger, D. Gan, G. Swallow, P. Pan, F. Tommasi, and S. Molendini. RSVP Refresh Overhead Reduction Extensions. RFC 2961, Internet Engineering Task Force, April 2001.

[99] C. Aoun and L.-N. Hamer. Potential Solutions to the Middle Box discovery problem. Internet Draft, Internet Engineering Task Force, May 2002. Work in progress.

[100] C. Aoun. Middle Box discovery integration solutions within the Midcom architecture. Internet Draft, Internet Engineering Task Force, June 2002. Work in progress.

[101] M. Karsten. *QoS Signalling and Charging in a Multi-service Internet using RSVP.* PhD thesis, Darmstadt University of Technology, July 2000.

[102] U. Roedig, M. Görtz, M. Karsten, and R. Steinmetz. RSVP as Firewall Signalling Protocol. Technical Report 6, KOM, December 2000.

[103] M. Karsten. KOM RSVP Engine Software. 2000. http://www.kom.e-technik.tu-darmstadt.de/rsvp.

[104] D. Reed. IP-Filter. http://coombs.anu.edu.au/avalon.

[105] K. Cho. ALTQ: Alternate Queueing for BSD UNIX . http://www.csl.sony.co.jp/person/kjc/kjc/software.html.

[106] U. Roedig, M. Görtz, M. Karsten, and R. Steinmetz. RSVP as Firewall Signalling Protocol. In *Proceedings of the 6th IEEE Symposium on Computers and Communications, Hammamet, Tunisia*, pages 57–62. IEEE, July 2001.

[107] M. Shore. The TIST (Topology-Insensitive Service Traversal) Protocol. Internet Draft, Internet Engineering Task Force, May 2002. Work in progress.

[108] Microsoft. Microsoft NetMeeting. Technical report, Microsoft, February 2002. http://www.microsoft.com/windows/netmeeting.

[109] Cisco Systems. Cisco Multimedia Conference Manager MMCM. http://www.cisco.com.

[110] U. Roedig, R. Ackermann, and R. Steinmetz. Evaluating and Improving Firewalls for IP-Telephony Environments. In *Proceedings of the 1st IP-Telephony Workshop (IPTel2000), Berlin*, number ISSN 1435-2702, pages 161–166. GMD-Forschungszentrum Informationstechnik GmbH, April 2000.

[111] U. Roedig, R. Ackermann, D. Rohrdrommel, J. Schlesinger, and R. Steinmetz. H.323 Proxy Call Dispatcher. Patent Registration, August 2000.

[112] U. Roedig, R. Ackermann, D. Rohrdrommel, J. Schlesinger, and R. Steinmetz. Firewall parser architecture for a given protocol. Patent Registration, April 2000.

[113] U. Roedig. KOMproxyd Software. 2001. http://www.kom.e-technik.tu-darmstadt.de/KOMproxyd.

[114] A. Moizard. The GNU oSIP library. http://www.gnu.org/software/osip.

[115] M. Zink, C. Griwodz, and R. Steinmetz. KOM Player - A Platform for Experimental VoD Research. In *Proceedings of the 6th IEEE Symposium on Computers and Communications*, pages 370–375, July 2001. ISBN 0-7695-1178-5.

[116] GigaPort. Videoconferencing achter de firewall. http://www.gigaport.nl/netwerk/access/doc/h323.

[117] M. M. Buddhikot, S. Suri, and M. Waldvogel. Space Decomposition Techniques for Fast Layer-4 Switching. In J. D. Touch and J. P. G. Sterbenz, editors, *Protocols for High*

Speed Networks IV (Proceedings of PfHSN '99), pages 25–41, Salem, MA, USA, August 1999. Kluwer Academic Publishers.

[118] International Telecommunication Union. Network grade of service parameters and target values for circuit-switched services in the evolving ISDN. *Recommendation E.721, Series E: Overall Network Operation, Telephone Service, Service Operation and human factors. Telecommunication Standardization Sector of ITU, Geneva, Switzerland*, 1999.

[119] European Telecommunications Standards Institute. End-to-End Quality of Service in TIPHON Systems; Part 2: Definition of speech Quality of Service (QoS) classes. *Draft, Telecommunications and Internet Protocol Harmonization over Networks, ETSI*, 2000.

[120] T. Eyers and H. Schulzrinne. Predicting Internet Telephony Call Setup Delay. In *Proceedings of the 1st IP-Telephony Workshop (IPtel 2000)*, Berlin, Germany, April 2000.

[121] P. Gupta and N. McKeown. Algorithms for Packet Classification. *IEEE Network*, March 2001.

[122] Netscreen. NetScreen 5000. http://www.netscreen.com.

[123] T. Varga. owdm. http://hsnlab.ttt.bme.hu/ varga/tools/owdm.tar.

[124] S. Ubik, V. Smotlacha, S. Saaristo, and J. Laine. Low-Cost Precise QoS Measurement Tool. *CESNET technical report number 7/2001*, June 2001.

[125] OpenH323. OpenH323 Call Generator. http://www.openh323.org.

[126] Empirix. Hammer.323 Call Generator. http://www.empirix.com.

[127] G. Varghese and A. Lauck. Hashed and Hierarchical Timing Wheels: Data Structures for the Efficient Implementation of a Timer Facility. *Operating Systems Review Special Issue: Proceedings of the Eleventh Symposium on Operating Systems Principles, Austin, TX, USA*, 21(5):25–38, November 1987.

[128] U. Roedig. KOMtraffgen Software. 2002. http://www.kom.e-technik.tu-darmstadt.de/KOMtraffgen.

Abkürzungen

ALTQ	Alternate Queueing
ASCII	American Standard Code for Information Interchange
ATM	Asynchronous Transfer Mode
COPS	Common Open Policy Service
CPU	Central Processing Unit
CTC	Counter/Timer Chip
DFN	Deutsches Forschungsnetz
DMZ	Demilitarisierte Zone
DoS	Denial of Service
FCP	Firewall Control Protocol
FTP	File Transfer Protocol
GPS	Global Positioning System
HFCI	H.323 Firewall Control Interface
HTTP	Hypertext Transfer Protocol
IDS	Intrusion Detection System
IETF	Internet Engineering Task Force
ILS	Internet Location Server
IO	input-output
IP	Internet Prococol
IPsec	IP Security
IPv4	Internet Protocol Version 4
IPv6	Internet Protocol Version 6
ISDN	Integrated Services Digital Network
IT	Information Technology

ITU-T	International Telecommunications Union - Telecommunications Standardization Sector
LC-RTP	Loss Collection RTP
LDAP	Light-weight Directory Access Protocol
MCU	Multipoint Control Unit
MEGACO	Media Gateway Control
MIDCOM	Middlebox Communication
NAT	Network Address Translation
NTPD	Network Time Protocol Daemon
PC	Personal Computer
PDD	Post Dial Delay
PDU	Protocol Data Unit
PPD	Post Pickup Delay
PSTN	Public Switched Telephone Network
QoS	Quality of Service
RAS	Registration, Admission, Status
RSIP	Realm Specific IP
RSVP	Resource Reservation Protocol
RTC	Real Time Clock
RTCP	RTP Control Protocol
RTP	Realtime Transport Protocol
RTSP	Real-Time Streaming Protocol
SDP	Session Description Protocol
SIP	Session Initiation Protocol
SMTP	Simple Mail Transport Protocol
SNMP	Simple Network Management Protocol
SPI	Security Parameter Index
TARP	Transient Addressing for Related Processes
TCP	Transfer Control Protocol
TSC	Time Stamp Counter
UDP	User Datagram Protocol
URL	Uniform Resource Locator
VoD	Video on Demand
VoIP	Voice over IP

VPN	Virtual Private Network
WWW	World Wide Web

Anhang A: Eigene Veröffentlichungen

Bücher

Stephan Fischer, Christoph Rensing, and Utz Roedig. Open Internet Security - Von den Grundlagen zu den Anwendungen. Springer Verlag, Heidelberg, Germany, January 2000. ISBN 3-540-66814-4.

Markus Schumacher, Utz Roedig, and Marie-Luise Moschgath. Hacker Contest. Springer Verlag, Heidelberg, Germany, To appear. ISBN 3-540-41164-X.

Zeitschriften

Markus Schumacher, Marie-Luise Moschgath, and Utz Roedig. Angewandte Informationssicherheit: Ein Hacker-Praktikum an Universitäten. Informatik Spektrum, 6(23), June 2000.

Ralf Ackermann, Utz Roedig, and Ralf Steinmetz. Entwicklung und Nutzung von IP-Telefonie Anwendungen auf Unix-Systemen. GUUG Nachrichten, 2000(2), pages 37-41, September 2000.

Utz Roedig, Ralf Ackermann, and Ralf Steinmetz. IP-Telefonie und Firewalls, Probleme und Lösungen. Praxis in der Informationsverarbeitung und Kommunikation (PIK), 1(24), pages 32-40. January 2001.

Patente

Utz Roedig, Ralf Ackermann, Dieter Rohrdrommel, Jürgen Schlesinger, and Ralf Steinmetz. Firewall Parser Architecture for a Given Protocol. Patent Registration EP00107854, April 2000.

Utz Roedig, Ralf Ackermann, Christoph Rensing, Dieter Rohrdrommel, Juergen Schlesinger, and Ralf Steinmetz. Distributed Firewall for Multimedia Applications. Patent Registration EP00113530, June 2000.

Utz Roedig, Ralf Ackermann, Dieter Rohrdrommel, Juergen Schlesinger, and Ralf Steinmetz. H.323 Proxy Call Dispatcher. Patent Registration DE10040463, August 2000.

Konferenzbeiträge

Abdulmotaleb El Saddik, Gunter Weiss, Heiko Straulino, Utz Roedig, and Ralf Steinmetz. Erfahrungen in der kooperativen Java-Entwicklung: Ein Praxisbericht. In Proceedings of the Workshop on Java in Telecommunications 1997, Darmstadt, Germany, May 1997.

Christoph Rensing, Ralf Ackermann, Utz Roedig, Lars Wolf, and Ralf Steinmetz. Sicherheitsunterstützung für Internet Telefonie. DuD-Fachbeiträge, Sicherheitsinfrastrukturen, pages 285-296, March 1999. ISBN 3-528-05709-2.

Christoph Rensing, Utz Roedig, Ralf Ackermann, Lars Wolf, and Ralf Steinmetz. VDMFA, eine verteilte dynamische Firewallarchitektur für Multimedia-Dienste. In Proceedings of the Kommunikation in verteilten Systemen (KiVS), Darmstadt, Germany, pages 144-157, March 1999. ISBN 3-540-65597-2.

Utz Roedig, Ralf Ackermann, Marc Tresse, Lars Wolf, and Ralf Steinmetz. Verbesserte Systemsicherheit durch Kombination von IDS und Firewall. DuD-Fachbeiträge, Systemsicherheit, pages 117-128, March 2000. ISBN 3-528-05745-9.

Utz Roedig, Ralf Ackermann, and Ralf Steinmetz. Evaluating and Improving Firewalls for IP-Telephony Environments. In Proceedings of the 1st IP-Telephony Workshop (IPTel2000), Berlin, pages 161-166, April 2000. ISSN 1435-270-2.

Utz Roedig, Ralf Ackermann, Christoph Rensing, and Ralf Steinmetz. A Distributed Firewall for Multimedia Applications. In Proceedings of the Workshop "Sicherheit in Netzen und Medienströmen", Berlin, pages 3-16, September 2000. ISSN 1431-472-X.

Christoph Rensing, Utz Roedig, Ralf Ackermann, and Ralf Steinmetz. A Survey of Requirements and Standardization Efforts for IP-Telephony-Security. In Proceedings of the Workshop "Sicherheit in Netzen und Medienströmen", pages 50-60, September 2000. ISSN 1431-472-X.

Ralf Ackermann, Vasilios Darlagiannis, Utz Roedig, and Ralf Steinmetz. Using DMIF for abstracting from IP-Telephony Signaling Protocols. In Proceedings of the Seventh International Workshop on Interactive Distributed Multimedia Systems and Telecommunication Services (IDMS 2000), Entschede, pages 104-115, October 2000.

Ralf Ackermann, Utz Roedig, Michael Zink, Carsten Griwodz, and Ralf Steinmetz. Associating IP data streams with user identities - enabling enhanced security, billing and copyright protection. In Proceedings of the Multimedia and Security Workshop at ACM Multimedia 2000, Los Angeles, pages 149-152, October 2000.

Ralf Ackermann, Markus Schumacher, Utz Roedig, and Ralf Steinmetz. Vulnerabilities and Security Limitations of current IP Telephony Systems. In Proceedings of the Conference on Communications and Multimedia Security (CMS 2001), Darmstadt, pages 53-66, May 2001.

Utz Roedig, Manuel Görtz, Martin Karsten, and Ralf Steinmetz. RSVP as Firewall Signalling Protocol. In Proceedings of the 6th IEEE Symposium on Computers and Communications, Hammamet, Tunisia, pages 57-62, July 2001.

Markus Schumacher and Utz Roedig. Security Engineering with Patterns. In 8th Conference on Pattern Languages of Programs (PLoP 2001), September 2001.

Martin Steinebach, Frank Siebenhaar, Christian Neubauer, Ralf Ackermann, Utz Roedig, and Jana Dittmann. Intrusion Detection Systems for IP Telephony Networks. In Proceedings of the Symposium Real Time Intrusion Detection Symposium, Portugal, May 2002.

Sonstiges

Utz Roedig. Firewalls and their Impact on Multimedia Systems. Multimedia Computing and Networking 2000. Panel Discussion "Security Firewalls and their Impact on Multimedia Systems", January 2000

Utz Roedig. KOMproxyd Software, 2001.
http://www.kom.e-technik.tu-darmstadt.de/KOMproxyd.

Utz Roedig. KOMtraffgen Software, 2002.
http://www.kom.e-technik.tu-darmstadt.de/KOMtraffgen.

Anhang B: Nachrichtenformate der Testapplikation

B.1 Nachrichtenformat des Signalisierungskanals

Die nachfolgenden Darstellungen erläutern die gültigen Nachrichten, die innerhalb des Kontrollkanals des Test-Protokolls verwendet werden können.

Header Beschreibung. Abbildung 67 beschreibt das Grundformat der Nachrichten; in den folgenden Abbildungen sind die Formate der verschiedenen möglichen Nachrichtentypen gegeben.

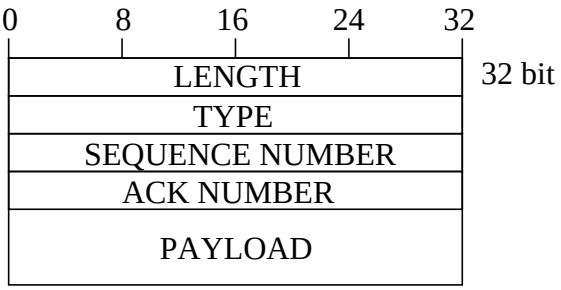


Abbildung 67: Test-Protokoll: Kontrollkanal Paket-Header

- *LENGTH*: Anzahl der 32-bit Worte, die in der Nachricht enthalten sind.
- *TYPE*: Eine Zahl, die den Typ der Nachricht kennzeichnet.
- *SEQUENCE NUMBER*: Jedes Paket, das in einer Session in die gleiche Richtung gesendet wird, muss eine eindeutige Sequenznummer verwenden. Beide Richtungen der Kommunikation können identische Sequenznummern verwenden.
- *ACK NUMBER*: Die Sequenznummer eines zuvor empfangenen Paketes, das bestätigt werden soll, oder Null wenn keine Bestätigung ausgeführt wird.
- *PAYLOAD*: Enthält die Informationen die durch die einzelenen Pakettypen transportiert werden.

HELO - TYPE 101. Diese Nachricht wird auf dem Signalisierungskanal verwendet, um die Kommunikation zu initiieren. Diese Nachricht wird zwischen Client und Server als erstes übermittelt und kann durch im Signalisierungspfad liegende Komponenten (Infrastrukturkomponeten) für ein Routing verwendet werden.

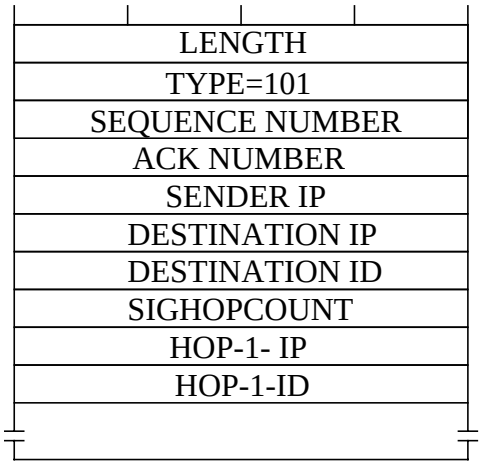


Abbildung 68: Test-Protokoll: *HELO*-Nachricht

- *SENDER IP*: Die IP-Adresse des Hosts, der die Nachricht generiert hat.
- *DESTINATION IP*: Die IP-Adresse des Zielsystems. Wenn diese Adresse unbekannt ist (z.B. in NAT-Szenarien), muss dieses Feld auf Null gesetzt werden.
- *DESTINATION ID*: Eine symbolische Kennzeichnung des Zielsystems. Diese Kennzeichnung muss verwendet werden, wenn das Feld *DESTINATION IP* auf Null gesetzt ist. Das Feld muss auf Null gesetzt werden, wenn es nicht

verwendet wird. Die Kennzeichnung kann von Infrastrukturkomponeten für ein Routing des Signalisierungskanals verwendet werden.

- *SIGHOPCOUNT*: Gibt die Anzahl der folgenden *HOP-IP/HOP-ID* Paare an. Dieses Feld muss zu Null gesetzt werden, wenn es nicht verwendet wird.
- *HOP-IP*: Die IP-Adresse des nächsten “Hop”, der im Signalisierungspfad verwendet wird. Dieses Feld muss zu Null gesetzt werden, wenn es nicht verwendet wird.
- *HOP-ID*: Eine symbolische Kennzeichnung des nächsten “Hop”, der im Signalisierungspfad verwendet wird. Diese Kennzeichnung muss verwendet werden, wenn *HOP-IP* Null ist. Dieses Feld muss zu Null gesetzt werden, wenn es nicht verwendet wird.

PLAY - TYPE 102. Diese Nachricht wird auf dem Kontrollkanal verwendet, um einen Medienstrom zu initiieren.

- *IP DEST ADDRESS*: Dieses Feld gibt die Ziel-IP-Adresse des angeforderten Medienstroms an.
- *IP DEST PORT*: Dieses Feld gibt den Ziel-Port des angeforderten Medienstroms an.

LENGTH
TYPE=301
SEQUENCE NUMBER
ACK NUMBER
IP DEST ADDRESS
IP DEST PORT

Abbildung 69: Test-Protokoll: *PLAY*-Nachricht

ACK - TYPE 103. Diese Nachricht wird auf dem Kontrollkanal verwendet, um den Erhalt einer Signalisierungsnachricht zu bestätigen. Diese Nachricht wird verwendet, wenn eine Bestätigung nicht als “piggyback” mit einer anderen Kontrollnachricht verschickt werden kann.

- *SEQUENCE NUMBER*: Die Sequenznummer in dieser Nachricht wird auf Null gesetzt, da Bestätigungen nicht selbst nochmals bestätigt werden können.

LENGTH
TYPE=102
SEQUENCE NUMBER = 0
ACK NUMBER

Abbildung 70: Test-Protokoll: *ACK*-Nachricht

STOP - TYPE 104. Diese Nachricht wird verwendet, um eine Verbindung zwischen den Kommunikationspartnern zu beenden.

LENGTH
TYPE=103
SEQUENCE NUMBER
ACK NUMBER

Abbildung 71: Test-Protokoll: *STOP*-Nachricht

B.2 Nachrichtenformat des Datenkanals

Die nachfolgenden Darstellungen erläutern die gültigen Nachrichten, die innerhalb der Medienkanäle des Test-Protokolls verwendet werden können.

Header Beschreibung. Abbildung 72 beschreibt das Grundformat der Nachrichten; in den folgenden Abbildungen sind die Formate der verschiedenen möglichen Nachrichtentypen gegeben.

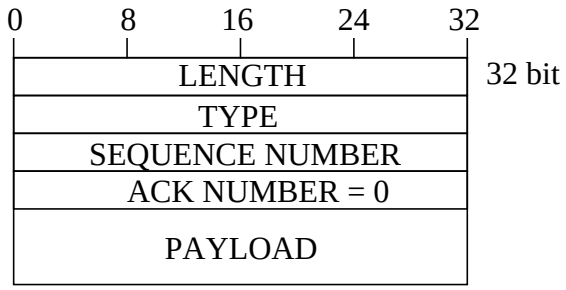


Abbildung 72: Test-Protokoll: Datenkanal Paket-Header

- *LENGTH*: Anzahl der 32-bit Worte die in der Nachricht enthalten sind.
- *TYPE*: Eine Zahl, die den Typ der Nachricht kennzeichnet.
- *SEQUENCE NUMBER*: Jedes Paket verwendet eine eindeutige Sequenznummer. Diese wird auf Client Seite generiert; das Paket wird auf Server-Seite reflektiert.
- *ACK NUMBER*: Pakete auf dem Medienkanal werden nicht bestätigt. Dieses Feld ist immer Null und existiert aus Kompatibilitätsgründen.
- *PAYLOAD*: Enthält die Informationen, die durch die einzelenen Pakettypen transportiert werden.

TEST - TYPE 501. Diese Nachricht wird für Testzwecke auf dem Medienkanal verwendet. Die Nachricht besitzt eine feste Größe von 400 byte.

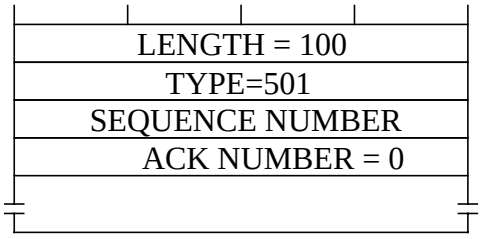


Abbildung 73: Test-Protokoll: *TEST*-Nachricht

DATA - TYPE 502. Diese Nachricht wird auf dem Kontrollkanal für Messungen verwendet. Dieses Paket wird auf Client-Seite generiert und auf Server-Seite reflektiert.

- *CLIENT_S TIME SEC*: Sendezeitpunkt auf Client-Seite (Sekunden).
- *CLIENT_S TIME USEC*: Sendezeitpunkt auf Client-Seite (Mikrosekunden).
- *CLIENT_R TIME SEC*: Empfangszeitpunkt auf Client-Seite (Sekunden).
- *CLIENT_R TIME USEC*: Empfangszeitpunkt auf Client-Seite (Mikrosekunden).
- *SERVER_S TIME SEC*: Sendezeitpunkt auf Server-Seite (Sekunden).
- *SERVER_S TIME USEC*: Sendezeitpunkt auf Server-Seite (Mikrosekunden).
- *SERVER_R TIME SEC*: Empfangszeitpunkt auf Server-Seite (Sekunden).
- *SERVER_R TIME USEC*: Empfangszeitpunkt auf Server-Seite (Mikrosekunden).

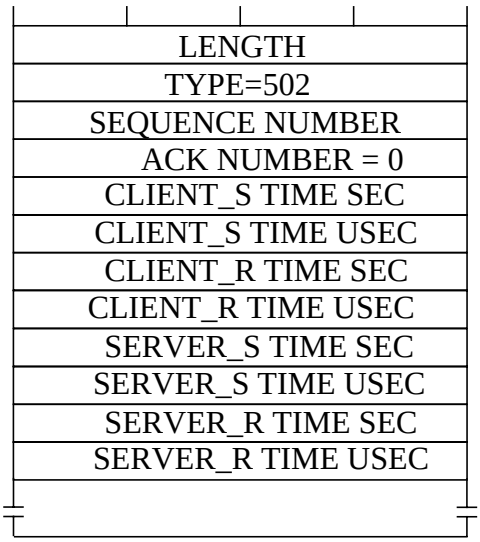


Abbildung 74: Test-Protokoll: *DATA*-Nachricht

Anhang C: Zusätzliche Darstellungen der Messergebnisse

Im Folgenden sind für die in Kapitel 7 durchgeführten Messungen zusätzliche Darstellungen der Messergebnisse gegeben.

C.1 Eichmessung

Sitzungsaufbaudauer. Darstellung der Sitzungsaufbaudauer inklusive Standardabweichung (Messung Abschnitt 7.5.2).

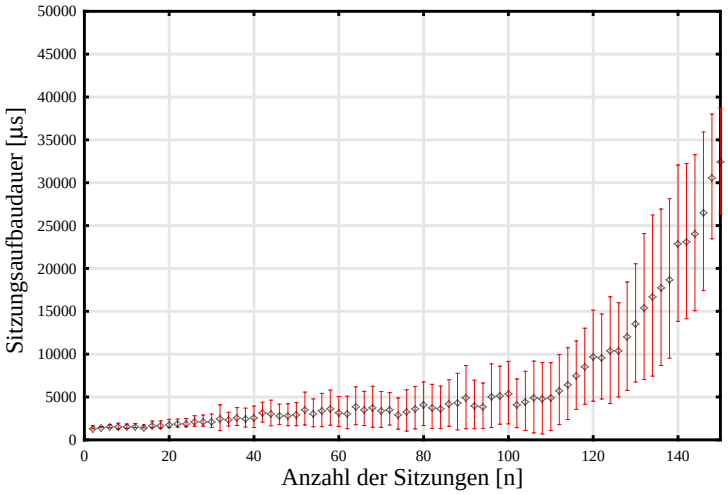


Abbildung 75: Eichmessung: Sitzungsaufbaudauer

C.2 Optimierung “nur Sitzungsaufbau”

Sitzungsaufbaudauer. Darstellung der Sitzungsaufbaudauer inklusive Standardabweichung (Messung Abschnitt 7.5.3).

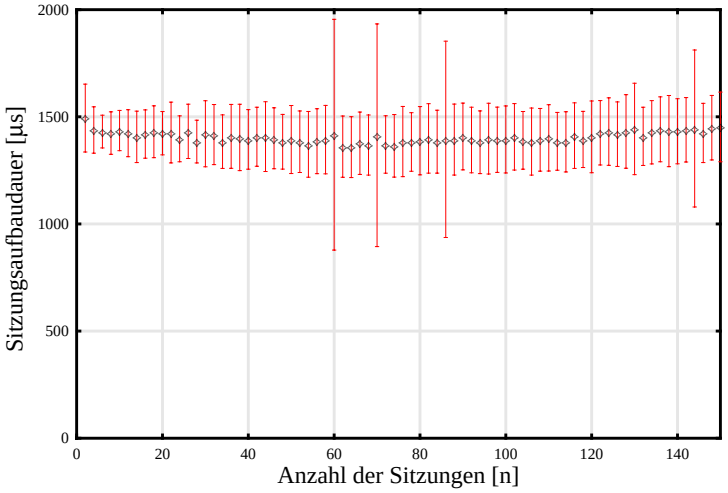


Abbildung 76: Eichmessung: Sitzungsaufbaudauer (optimiert, *nur Sitzungsaufbau*)

C.3 Optimierung “Kernel Stamping”

Sitzungsaufbaudauer. Darstellung der Sitzungsaufbaudauer inklusive Standardabweichung (Messung Abschnitt 7.5.3).

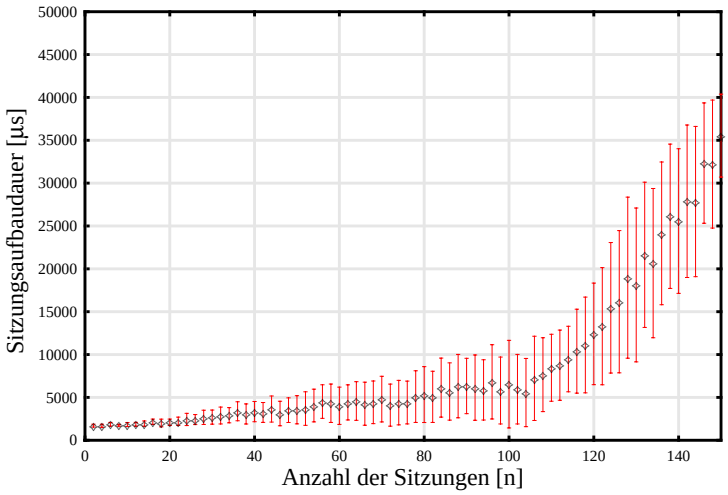


Abbildung 77: Eichmessung: Sitzungsaufbaudauer (optimiert, *Kernel Stamping*)

Sitzungsaufbaudauer. Darstellung der Zusammensetzung der Sitzungsaufbaudauer $T_s = T_a + T_{b1} + T_{b2}$ (Messung Abschnitt 7.5.3).

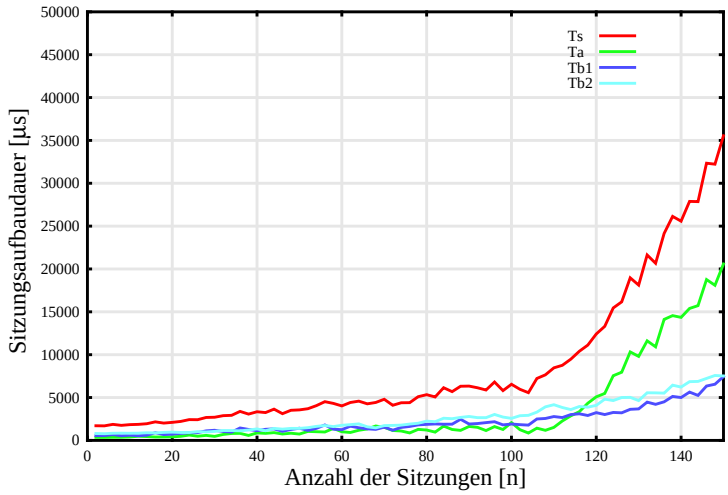


Abbildung 78: Eichmessung: Sitzungsaufbaudauer (optimiert, *Kernel Stamping*)

C.4 Leistungsfähigkeit verteilter Firewall-Architekturen

Sitzungsaufbaudauer. Darstellung der Sitzungsaufbaudauer der Eichmessung inklusive Standardabweichung (Messung Abschnitt 7.7.2)

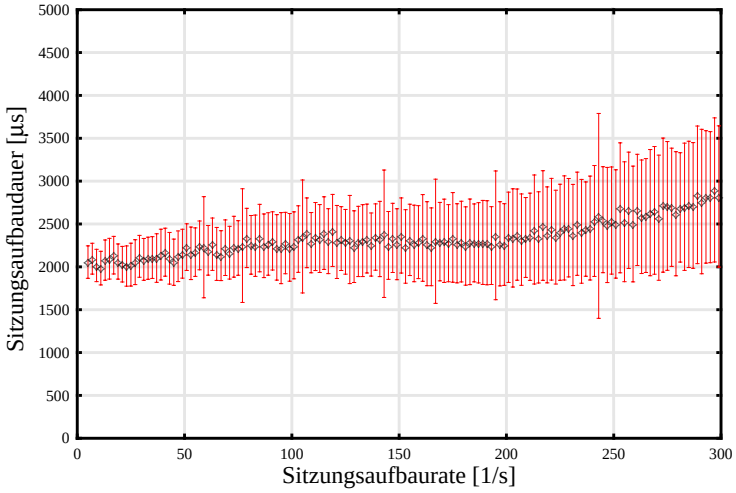


Abbildung 79: II: Sitzungsaufbaudauer (Eichmessung)

Index

Z

Lebenslauf

Name:	Utz Roedig
Geburtstag:	06.07.1972

Schule und Studium

1977 - 1982	Grundschule Lippertsreute
1982 - 1988	Gymnasium Überlingen
1988 - 1991	Berufliches Gymnasium Überlingen Abschluß: Abitur 06/1991
1991 - 1997	Studium Elektrotechnik, Technische Universität Darmstadt Abschluß: Diplom-Elektrotechniker, 7/1997

Berufliche Tätigkeit

08/1997 - 03/1998	Wissenschaftlicher Mitarbeiter im Hochschulrechenzentrum Darmstadt
seit 03/1998	Wissenschaftlicher Mitarbeiter im Fachbereich Elektrotechnik und Informationstechnik der Technischen Universität Darmstadt